FAI Designs Inc.

REAL-TIME EXPERT SYSTEM

R T E S

USER MANUAL

C2002 – L2002
12th Edition - July 2005

NOTICE

The use of the software packages described in this manual is governed by the terms and conditions set out in the software license agreement between FAI Designs Inc. and the licensee.

This manual is not to be copied or reproduced, by any means, in whole or in part, without the express authorization of FAI Designs Inc.

TABLE OF CONTENTS

CHAPTER ONE - REAL-TIME EXPERT SYSTEM

What is the Real-Time Expert System 1-1
How does the Real-Time Expert System work 1-1
The major features of the Real-Time Expert System 1-3
The Real-Time Expert System and Artificial Intelligence 1-3
How to use this manual 1-4
Safety warning 1-5

CHAPTER TWO - STARTUP AND INSTALLATION PROCEDURE

Install the RTES program on your hard disk 2-1
Running RTES 2-2
Handling RTES knowledge base files 2-3
Interfacing with the real world 2-3
Starting RTES automatically on power up 2-4

CHAPTER THREE - KEYBOARD

Extended Character Set 3-1
Line Editing 3-1
Function keys 3-2
- during On-Screen Database Edit Mode 3-4
- during Customized Page Editing Mode 3-4
Case Significance 3-6
Separators 3-6

CHAPTER FOUR - HOW TO DEVELOP A KNOWLEDGE BASE

Getting started 4-1
Example application 4-1
Defining the variables 4-1
Building a display page 4-4
Defining the rules 4-6
Testing and debugging the rule 4-7
Interfacing with the real world 4-8
Loading the OPTOMUX driver 4-8
Defining the alerts 4-9

CHAPTER FIVE - DATABASE

Database 5-1
Variables 5-1
Numeric Variables 5-2
Binary Variables 5-2
How to display the variables 5-2

Binary Inputs 5-3
Binary Outputs 5-4
Analog Inputs 5-4
Analog Outputs 5-6
Converters using other than 12 bit resolution 5-6
On screen editing 5-6
Starting a database on-screen editing session 5-7
Special keys 5-7
Ending a database editing session 5-8
Valid entries 5-8
Database editing by commands 5-9
Controlling the variables 5-11
- by command 5-12
- by function key 5-12
- on-line calculator 5-13
- the LET command 5-13
Override 3-14
The "E" column 3-14

CHAPTER SIX - RULES
General Definitions 6-1
How to define or edit a rule 6-3
How to remove a rule 6-4
Structure of the Expression 6-4
Table of Allowed Operators 6-5
Precedence of the Operations 6-6
How Operators are Used in Expressions 6-6
Arithmetic operators 6-6
Relational operators 6-7
Logical operators 6-8
Delay operators 6-8
Automatic reset time delay 6-9
Cumulative timer 6-9
Variable indexing 6-10
Integer 6-10
Fault handling 6-10
"DEBUG" feature 6-11
Case significance 6-11
Separators 6-11
The inference engine 6-11
Examples and Typical Applications 6-12

CHAPTER SEVEN - SPECIAL CONTROL ALGORITHMS
Loop control 7-1
Defining a loop 7-1
Displaying a loop 7-2

Velocity type PID loop control 7-2
Adaptive control applications 7-3
Cascading loop control applications 7-3
Cascading registers 7-3
Defining a register cascade 7-4
Arc cosine 7-5
Command lists 7-5
Executing external code 7-6
Sample external code 7-7
Removing a function from the knowledge base 7-9
Bit handling facilities (FDEC & BITS) 7-10

CHAPTER EIGHT - TASKS
Overview 8-1
How to Create a Task 8-2
How to view or modify an existing task 8-3
Task directives 8-3
Syntax 8-4
Indexing with the "LET" Task Directive 8-5
File handling 8-5
How to Control the Execution of a Task 8-8
ONEND statement 8-9
One step at a time task execution 8-10
Missing Instructions 8-10
Monitoring Task Status 8-11
How to Change the Name of a Task 8-11
How to Remove a Task from the System 8-11
Note regarding the use of the "X" command 8-12
Autoexec 8-12
Using tasks and rules 8-12
Task synchronization 8-12
Using tasks efficiently 8-13
How to print a task listing 8-14
Disk resident tasks 8-15
Report generator with tasks 8-16
Text handling directives 8-17

CHAPTER NINE - DISPLAY GENERATOR
Overview 9-1
How to Create a new Display 9-1
How to modify a Display 9-2
The display editor 9-2
Changing the color 9-3
How to change the name of a display 9-4
How to show variables on a display 9-4
NUMBER 9-5

LZI 9-6
REAL 9-6
HGRAPH 9-6
VGRAPH 9-7
BINARY 9-7
LIST 9-7
TEXT 9-9
Display a variable as a color attribute 9-10
HCA 9-10
VCA 9-11
Error conditions 9-11
Modifying a Variable Specification 9-11
Space Restrictions 9-12
Deleting a Variable from a Page 9-12
Terminating a Display Edit Session 9-13
Examples 9-13
List of display pages 9-14
User defined screen title 9-15
User defined operator's instructions 9-16
User defined function keys 9-16
Control of operator accessible variables 9-17
Disk resident displays 9-18
Recall of last display 9-19
Extended help facility 9-19
Creating an extended help screen 9-20
Using the extended help 9-20
Default help file 9-21
Context dependent help files 9-21
Multi-screen help files 9-21

CHAPTER TEN - TIME OF DAY SCHEDULER
Time of day scheduler 10-1
Defining a schedule event 10-1
Display scheduled events 10-1
Print scheduled events 10-2

CHAPTER ELEVEN - TREND RECORDING
Overview 11-1
Typical set up 11-1
Events file 11-2
Other system information associated with logging 11-3

CHAPTER TWELVE - THE ALERT SYSTEM
Overview 12-1
How to define an alert 12-1
Alert acknowledge 12-2

Alert status variables 12-3
Display current active alerts 12-3
Logging alert messages on the printer 12-3
Logging alerts on disk 12-4
Alert historian 12-5
Alert extension facility 12-6
Creating an alert extension screen 12-6
Using the alert extension 12-6
Context dependent alert extension files 12-7
Multi-screen alert extension files 12-7
Change of state logger 12-7

CHAPTER THIRTEEN - BATCH COMMAND FACILITY
Overview 13-1
Batch file format 13-1
Batch file execution 13-2
Automatically created batch command files 13-3
name.FMT file format 13-4
Default FILE argument 13-5

CHAPTER FOURTEEN - REPORT GENERATOR
Overview 14-1
Imbedded Variables 14-1
Print on Condition 14-2
The PRINT command 14-3
Printing the knowledge base 14-3
Printing in progress flag 14-4
External Print Font Definition 14-4
Abort current print command 14-6

CHAPTER FIFTEEN - COMMUNICATIONS INTERFACE
Introduction 15-1
Loading a RTES driver 15-2
Configuration files 15-3
Register index versus register number 15-4
Parameters 15-4
Baud rate 15-5
Time-out constant 15-5
Modem support 15-6
Serial port and IRQ 15-6
Block enable option 15-7
Port failure 15-7
I/O port address 15-7
Real-Time clock refresh 15-8
I/O assignment 15-8
Commonly used device drivers 15-11

PAMUX 15-11
PCLAB 15-11
SIXNET 15-13
OPTOMUX 15-16
Mitsubushi PLC 15-17
Allen Bradley PLC 15-18
MODICON PLC 15-20
OMRON PLC 15-21
Siemens PLC 15-22
Texas Instrument 500 Series 15-23
Protocol analyzer 15-25
Troubleshooting communication problems 15-25
Performance tips 15-27
Real Time Clock refresh 15-28
Network drivers 15-28
Host 15-29
Slave 15-30
Note re data in distributed systems 15-31
Using network drivers with other systems 15-31
RTES - OPTOMUX network emulator 15-34
Using external code with RTES 15-36
Writing your own device driver 15-40

CHAPTER SIXTEEN - HIGH RESOLUTION GRAPHICS
Overview 16-1
Screen resolution 16-2
Graphics commands 16-3
Syntax 16-4
Screen dump 16-7
Installation 16-9
Example 16-10
Get screen utility 16-11

CHAPTER SEVENTEEN - CONFIGURATION PARAMETERS
Reserved variable names 17-1
Configuration file 17-3
User level 17-4
Machine type 17-4
Printer control 17-5
Example configuration file 17-5
System configuration
Standard RTES 17-6
RTES-OEM 17-7
RTES-MAX (Jumbo) 17-8

CHAPTER EIGHTEEN - RTES COMMANDS

A	Display an analog input definition page	18-1
ACK	Acknowledge the most recent alert	18-1
AEX	Display an alert extension file	18-1
ALERT	Define an alert state	18-2
ARC	Calculate the Arc Cosine	18-2
BUILD	Create a new customized display page	18-3
CLS	Clear screen	18-3
COLOR	Fill an area of the screen	18-3
CONF	Load in new configuration for RTES	18-4
COPY	Copy a file	18-4
CTEXT	Create a text from a template file	18-4
DATE	Set the calendar	18-5
DEBUG	Set on or off the dynamic expanded rule feature	18-6
DEC	Set the default number of digits after the decimal point	18-6
DEL	Delete a file	18-6
DOC	Open/Close disk file to substitute for printer	18-7
EDIT	Enter the edit mode	18-8
END	Terminate an active task	18-8
ERASE	Remove a task from the system	18-8
EU	Define engineering unit	18-9
EXEC	Define name as a command list	18-9
EXT	Associate name with external code	18-9
FILE	Create a command file	18-10
FPD	Dump a display page to disk	18-11
FREE	Free a variable from being evaluated	18-11
FTASK	Load a task from a disk file	18-11
FTEXT	Load a variable definition file	18-12
GET	Execute a command file	18-12
HALT	Suspend execution of a task	18-13
HLP	Display an extended help file	18-13
HOME	Return to command mode	18-13
I	Display a digital input definition page	18-14
ID	Get the RTES serial number	18-14
L1	Turn On/Off the protocol monitor for PORT1	18-14
L2	Turn On/Off the protocol monitor for PORT2	18-15
L3	Turn On/Off the protocol monitor for PORT3	18-15
L4	Turn On/Off the protocol monitor for PORT4	18-15
LET	Assign a calculated value to a variable	18-15
LINE	Draw a line in pixel mode	18-16
LIST	Display the current active alerts	18-17
LOC	Locate a variable by name	18-17
LOOP	Define a loop	18-17
M	Turn on/off the inference engine monitor	18-18
NAME	Assign a name to a variable	18-18
NOTE	Create/edit a text file	18-19

O	Display a digital output definition page	18-19
OPEN	Open a task for editing	18-20
PA	Print the analog input definitions	18-20
PAGE	Display the names of the display pages	18-20
PAGE	Display a page	18-20
PAH	Print alarm history	18-21
PD	Dump a display page to disk	18-21
PDEF	Install printer definition file	18-21
PE	Print the rules currently in the knowledge base	18-22
PGDN	Next page	18-22
PGUP	Previous page	18-22
PI	Print the digital input definitions	18-22
PID	Define a velocity type PID loop	18-23
PL	Load a display page from disk	18-23
PO	Print the digital output definitions	18-24
PR	Print the register definitions	18-24
PRINT	Print a report	18-24
PRTSC	Print the content of line 4 to 20 of the screen	18-25
PS	Print all scheduled events	18-25
PT	Print a task listing	18-25
PULSE	Pulse an output or register	18-26
PY	Print the analog output definitions	18-26
R	Display a register definition page	18-26
RANGE	Define analog input range	18-27
REN	Assign a new name to an existing task	18-27
RULE	Display an existing rule	18-27
RUN	Initiate or resume execution of a task	18-28
SAVE	Save revised files on disk	18-28
SCHED	Define/Recall a scheduled event	18-28
SET	Set the value of a variable to a constant	18-29
SHIFT	Define/Recall a cascade of registers	18-29
SIZE	Display the memory used and available	18-30
START	Set a variable to 1 or ON	18-30
STEP	Make a task execute the next instruction	18-31
STOP	Set a variable to 0 or OFF	18-31
STP	Abort current print command	18-31
TASK	Display the status of the tasks	18-32
TASK	Create a task and open it for editing	18-32
TEXT	Assign a text descriptor to a variable	18-32
TEXT	Copy a text from another variable	18-33
TIME	Set the current time	18-33
TRACE	Display alarm history	18-33
TREND	Display the trend table	18-34
TS	Display status of all tasks	18-34
TXT	Display a text in pixel mode	18-34
TYPE	Define analog input conversion type	18-35

VAR	Display a variable in pixel mode	18-35
XFILE	Create a batch file	18-36
XP	Display an extended page	18-36
Y	Display an analog output definition page	18-37

APPENDIX A - Application examples

Domestic hot water control	A-1
Heat pump control	A-4
Fire alarm auxiliary system	A-6
Batch heating	A-10
Statistical quality control	A-13
Data averaging	A-17
A data Logger	A-20

APPENDIX B

Procedure to transport a knowledge base to a different version of RTES.

CHAPTER ONE - REAL-TIME EXPERT SYSTEM

WHAT IS THE REAL-TIME EXPERT SYSTEM

The Real-Time Expert System combines 'rule based' knowledge representation with data acquisition and control techniques. It allows the control and monitoring of industrial and/or laboratory equipment using a collection of rules that define the intended functions and operations.

The Real-Time Expert System directly connects to an industrial data acquisition system that will provide data from the instruments monitoring measurable quantities (temperature, pressure, etc.),. Using the rules contained in the knowledge base, the Real-Time Expert System will then make certain decisions. It could then start or stop a motor, increase or decrease the opening of a modulating valve, etc.).

Rules may be mathematical calculations, logical relationships or combinations of both.

The Real-Time Expert System includes all the "tools" to incorporate the rules into a 'knowledge base', maintain the knowledge base and make it control and monitor the equipment.

HOW DOES THE REAL-TIME EXPERT SYSTEM WORK

A database consists of information stored in an organized manner. The RTES database is a list of variables. Every variable has a name, a current value and a text. The name and text are defined by the user. These variables are an image of the process being controlled.

In addition, RTES also has a 'knowledge base'. It consists of RULES that explain how to calculate the values of the variables under given circumstances. These calculations are performed by a data driven inference engine.

When a variable changes, any rule affected by that change is 'triggered'. The calculation described by the rule takes place and the result is assigned to the variable affected. This may cause some other rules to be triggered. But, a calculation takes place only if the rule that describes it is triggered.

This process is called 'forward chaining'.

(c)1992-2005 FAI Designs Inc.

RTES creates its own multitasking environment that allows communication, display, inference engine, etc. to run concurrently.

RTES includes a built-in data base and rule editor. Development and maintenance of an application may take place while it is actually running that application. This allows a more effective method of developing the application, since you can test it and correct it as the process goes on.

Initially, it would be wise however, not to connect RTES to your process until you have validated the critical aspects of your operation. To that effect, RTES lets you simulate your process. Using keyboard commands such as SET, START, STOP, you may force the variables to change as if they were connected to the equipment.

When developing a knowledge base, you first define the variables (facts). Each variable is assigned a name and may represent a quantity or a list in which each element is a text. For instance, the variable "LEVEL" may represent a number. The variable "MODE" may represent a list consisting of the elements "MANUAL, AUTOMATIC, SEMI-AUTOMATIC".

Then you define the rules that will establish the relationships between the facts. These rules consist of arithmetic and logical expressions that the Real-Time Expert System will use to deduce the value of the conclusion facts from the known facts.

For instance, the rule "PUMP=AUTOMATIC and LEVEL<23" will cause the variable PUMP to be TRUE when the variable AUTOMATIC is TRUE and the variable LEVEL is less than 23. This rule will establish that a pump (connected to the binary output PUMP) will be energized when AUTOMATIC is true (this might be a binary input or an internal variable) and the analog input LEVEL, connected to a level transducer, indicates a value less than 23.

In order to simplify the use of the knowledge base, you may then design display pages that will present the application appropriately.

These customized displays serve as an interface between the operator and the process being controlled. They will provide status information, flag abnormal conditions, as well as allow the operator to change set points, operating modes or cause certain activities to take place.

(c)1992-2005 FAI Designs Inc.

THE MAJOR FEATURES OF THE REAL-TIME EXPERT SYSTEM

In addition to creating the proper environment for the definition and effective use of a knowledge base, the Real-Time Expert System incorporates several features that make it an easy to learn, easy to use, complete process control system. The Real-Time Expert System includes

* On line definition and editing of variables (facts) and rules using full screen editing techniques.

* On line creation and modification of dynamic display pages.

* On line logical programming editor.

* Conversion of analog data to engineering units.

* Concurrent execution of logical programs.

* Time of day scheduler.

* Alarm generator.

* PID loop control algorithm.

* Trend recording.

* Report generator.

* Cascading Registers.

* Change of State Logger.

THE REAL-TIME EXPERT SYSTEM AND ARTIFICIAL INTELLIGENCE

It would be hard to decide whether or not the Real-Time Expert System is an AI product, since there are many different definitions of such products, and there is certainly no unanimity on the subject. It is clear that the question first posed in 1950 ("Can machines think?"), is still

unanswered. It is also obvious that more and more, we tend to confuse intelligence with knowledge and that we become comfortable with the concept that machines that are knowledgeable, are considered intelligent. Maybe this is what we want, or need, for practical reasons. In this optic, the Real-Time Expert System is certainly headed in the right direction.

After all, if a machine were to act exactly like an intelligent human being, it might not be possible to get an answer from it, when you asked a question.

In addition to reacting automatically to changes, the Real-Time Expert System exhibits many of the features generally associated with AI systems. It is capable of performing symbolic computations, it may represent variables as numerical as well as text strings, it allows the use of sophisticated heuristic reasoning, it includes the mechanisms for the solution of incompletely stated problems.

But the Real-Time Expert System is not a programming language. There is no such thing as compiling a program, or linking modules, or any activity traditionally associated with software development, when using the Real-Time Expert System.

It is therefore ideally suited for the development of expert control algorithms by non-computer people and offers a high degree of friendliness to the end users, with very little training. We see the Real-Time Expert System as a practical solution to the problems created by the ever increasing complexity of instrumentation and process controls.

RTES was first introduced in 1985. It has since saved millions of dollars to its users by offering a low system development costs and enhancing productivity. We are sure that it will continue to find many more useful applications that justify its existence, we hope that it will inspire and encourage the development of other similar products that will help humanity cope with the vast amount of knowledge that will be available in the centuries to come.

HOW TO USE THIS MANUAL

If you are using the Real-Time Expert System in conjunction with an existing knowledge base, all you really need to know is how to use some of the function keys, specifically the ones associated with the control of variables, in addition to <Pg Up> and <Pg Dn> to find your way through the displays. You will find all the information in the KEYBOARD section. The knowledge base should include all the displays and assistance you need to operate the system.

If you intend to develop or maintain a knowledge base - containing one or more applications - then you should read the complete manual.

Refer to the COMMANDS section of this manual to get complete information on the use of each specific command. After you have used the commands a few times, the on screen HELP facility (<Alt-H>) will be sufficient to refresh your memory.

SAFETY WARNING

At some point, you will want to connect your computer to the process or machinery that it will be controlling. This is accomplished with a physical connections between your computer and the equipment. RTES will need a device driver to talk to your data acquisition and control subsystem.(See the DEVICE DRIVERS section of this manual)

When RTES is connected to a data acquisition and control subsystem, your application will activate the electrical circuitry and operate the moving parts of the system. Make sure you have properly tested your application (rules, tasks, etc.) and provided proper emergency shutdown before you actually apply power to your process or machinery.

If you must make changes to your application while on line, make sure you will not temporarily create an unsafe situation.

CHAPTER TWO
INSTALLATION AND START UP PROCEDURE
HARD DISK

This procedure of assumes that your system has a floppy disk drive A: and a hard disk C: and that DOS (version 2.11 or more recent) is installed on drive C:.

In the following description, <cr> represents the key on your keyboard marked 'CR' or 'Enter'.

INSTALL THE RTES PROGRAM ON YOUR HARD DISK

1. Boot-up your system.

2. Create a directory on drive C:. Call that directory RTES.

    MD\RTES<cr>

3. Make the directory your default by typing

    CD\RTES<cr>

4. Place the RTES original product diskette in drive A: and close the drive. Install the License block on the printer port of your computer.

5. Copy the relevant files from the RTES diskette to your new directory using the DOS command

COPY A:name<cr>

The following files are usually present in the RTES diskette:

* RTES.HLP the RTES help facility

*RTES.CNF a starter RTES configuration file that will allow you full access privilege. Later on, you will modify RTES.CNF to suit your own requirements.

* KTCTB.BIN used by RTES to linearize thermocouple analog inputs.

*READ.ME a text file you may print or type, giving the most recent information about the

version of RTES you are going to use (may not exist).

* EGA.EXE driver for EGA/VGA screens in pixel mode.

*CGA.EXE driver for CGA screens in pixel mode.

*HGA.EXE driver for Hercules displays in pixel mode.

* device drivers (*.EXE)

*RTES.EXE the RTES root program.

13. Remove the RTES diskette from drive A: and put a 'write protect' sticker on the notch to make sure that you will not accidentally erase or overwrite its content. You will have to remove this sticker only if you want to install RTES on another system. At this stage, you do not need your RTES original diskette any longer, until such time when you want to install it on another system or un-install it from this system.

Please store the diskette in a safe place, away from dust, humidity and extreme temperatures

RUNNING RTES

Your directory RTES is now ready to run RTES, with the following procedure.

1. Make sure the directory you want to use is the default by typing

CD\RTES<cr>

2. Start up RTES by typing

RTES<cr>

3. If you are a first time user of RTES, you may want to proceed with the sample session "How to create a knowledge base using RTES", provided with your manual.

TO EXIT FROM RTES BACK TO DOS, HOLD THE <Ctrl> KEY AND PRESS THE <Home> KEY. RTES saves its knowledge base in the current directory before returning your system to

DOS.

HANDLING RTES KNOWLEDGE BASE FILES

RTES creates and maintains a series of files in your current directory, called OLEX.NAM, OLEX.XRF, OLEX.DAT, OLEX.LNK and OLEX.PAG. The extended version (-MAX) also uses OLEX.DAX.

When RTES starts up, it automatically loads the content of these files, if they exist. RTES creates these files as needed.

If you wish to save away, a knowledge base you have created, you may copy these OLEX files (On Line EXpert), on a floppy. To do so, exit from RTES, place a formatted floppy in drive A: and type

COPY OLEX.* A:<cr>

If your application also includes your own help files (.HLP), alarm extension files (.AEX) and report files, be sure to copy them as well.

The OLEX files may be copied back in your current directory, but remember that they will overwrite any OLEX files that may be in that directory.

You may remove the OLEX files from your current directory, by typing

DEL OLEX.*<cr>

When you start up RTES, it will create a new set of OLEX files.

INTERFACING WITH THE REAL WORLD

The interface drivers with various I/O subsystems and PLC.s are included with your RTES package. Interface drivers with other devices are available and may be copied to your directory.

To use an interface driver, you must:

A. Create a file called PORTx.CNF (x being the port number), using your word processor or the RTES 'NOTE' facility , to include the communication port characteristics and the I/O assignments according to the instructions included in the appropriate section of this manual for the driver you are using.

B. Load the driver in memory BEFORE you run RTES by typing

driver_name p<cr>

where :driver_name is the name of the driver you want to use.

p is the (optional) port number (1 or 2) that you are connecting your I/O subsystem to.

For example, if you are using OPTOMUX on COM2, type

OPTOMUX 2<cr>

STARTING RTES AUTOMATICALLY ON POWER UP

It is often a good idea to arrange for RTES to start automatically, when the computer is powered up or at recovery from a power failure.

To do so, you must create an AUTOEXEC.BAT in the root directory of your C: drive. Make sure that there is no diskette left in A: drive, otherwise DOS will try and boot from A:.

The format of such AUTOEXEC.BAT might be for example:

```
PATH C:\
CD\RTES          ;or any directory you have set up
OPTOMUX 2        ;assuming you use OPTOMUX on COM2
PAMUX            ;assuming you also use PAMUX
RTES
```

CHAPTER THREE - KEYBOARD

EXTENDED CHARACTER SET

RTES allows the use of all printable characters. This includes upper and lower case letters, numerals, punctuation and other special symbols as shown on the keys. The following characters are exceptions:

shift-[displays the french 'e accent grave'.
shift-] displays the french 'e accent aigue'.
shift-\ displays the french 'c cedille'.

In addition, some characters are generated by using the 'ctrl' key in lieu of 'shift'. This includes:

√	square root	**<ctrl-S>**
≥	greater than or equal	**<ctrl-G>**
≤	less than or equal	**<ctrl-L>**

The above characters are changed to 'sqr', 'gte' and 'lte' when they are sent to the printer.

Accents (for foreign languages) are handled using the 'dead key' method. The accent is typed first, but is not echoed, then the vowel is typed which causes the complete character to be displayed. Should the character typed after the dead key be incompatible with the accent, the dead key is ignored and the character is displayed. At print time, RTES sends to the printer the vowel followed by a backspace (ASCII 08), then followed by the accent.

LINE EDITING

The keys used for line editing are:

<Ins> Toggle between character insert and replace mode. The bottom right hand of the screen indicates the current status with a 'I' or 'R', respectively.

<Backspace> Move content of the current input line one place to the left, from the cursor position.

<Del> Move content of the current input line one place to the left, from the position immediately to the right of the cursor.

<Arrow left> Move the cursor one position to the left.

<Arrow right> Move the cursor one position to the right.

<End> Move cursor to the end of the line, or to the beginning if it was at the end.

This feature may be used to effectively reduce the amount of typing required to operate RTES.

For instance, a previous command line may be modified into a new command by changing a few characters.

FUNCTION KEYS

Some keystrokes perform specific functions. These include the labelled keys and other character keys used in conjunction with the 'alt' in lieu of 'shift'.

Most of these functions will vary depending on what is executed at the time.

Function keys **<F1>** to **<F10>** may be re-assigned to execute various user defined functions while a user defined page is displayed. See the chapter on user interface customization for details.

The following keys are active at all times:

<alt-H> Display the on-line help

<ctrl-Home> Write changes to disk and return to operating system.

<Prt Sc> Print active portion of the screen (lines 4 to 20).

RTES is normally in 'command' mode, unless editing of some sort is performed. The command mode is characterized by the cursor being located on the second line of the screen. In this mode, the following function keys are defined.

<cr> End of input. Erase everything to the right of the cursor and submit the type line (command) to the command interpreter.

<Pg Up> Display previous page of a multi-screen display.

<Pg Dn> Display next page of a multi-screen display.

<Arrow up> Select previous variable on screen for control.

<Arrow down> Select next variable on screen for control.

<Tab> Same as <Arrow down>.

<Home> Cancel selection of variable.

<F7> Decrement or invert currently selected variable.

<F8> Increment or invert currently selected variable.

<Esc> Display extended help screen for selected variable or the default extended help screen if no variable is selected.

<Alt-X> Display alert extension screen for selected alert condition.

Most command words are generated by a single keystroke. If the command does not require an argument, then the keystroke actually executes the command. If an argument is required, the command word is displayed followed by a space (separator) and the cursor is positioned where the rest of the command is to be typed in.

The following keys, generate a command word:

<alt-A> ALERT
<alt-F> FREE
<alt-O> OPEN
<alt-R> RULE
<alt-S> SCHED
<alt-P> PAGE
<F1> START
<F2> STOP
<F3> SET
<F4> PULSE
<F5> RUN
<F6> HALT
<F9> TEXT

The following keys cause the command word to be displayed and the command to be executed:

<alt-D> DEBUG
<alt-E> EDIT
<alt-L> LIST
<alt-T> TASK
<F10> ACK

FUNCTION KEYS DURING ON-SCREEN DATABASE EDIT MODE

When variable definition pages are displayed, the screen editing feature may be entered using the command EDIT (or <alt-E>). The following keys then assume a special role in assisting in the database editing process.

<Home> End editing session and return to command mode.

<cr> Enter current field in the database.

<Pg Up> Previous page in same point category.

<Pg Dn> Next page in same point category.

<Arrow up> Previous line on page.

<Arrow down> Next line on page.

<Tab> Change field.

FUNCTION KEYS DURING CUSTOMIZED PAGE EDITING MODE

A new customized page may be created using the command BUILD. An existing customized page may be edited by using the EDIT (or **<alt-E>**) command while the page is on the screen. In either case the page editing mode is activated and the following keys have a special significance:

<F1> change the foreground color

<F2> change the background color

<F3> return to the default (white on black)

<Home> Terminate page edit mode and return to command mode.

<Arrow up> Move cursor one line up

<Arrow down> Move cursor one line down.

<cr> Move cursor to beginning of next line.

<End> Toggle DRAW MODE on/off.

With DRAW MODE on :

<F7> Select next pattern

<F8> Select previous pattern

Use the cursor movement arrow keys to draw with the selected pattern.

<alt-N> Allow editing of the page Name. The cursor returns to its original position on the screen when <cr> is pressed to enter the edited name or when <Home> is pressed, in which case the old name is restored.

<alt-R> Remove variable at current coordinates. RTES displays the variable and requests to type <C> (upper case) for confirmation.
If any other key is pressed, the command is ignored.

<alt-V> Define or edit variable at current coordinates. If a variable exists at the coordinates, RTES displays its specification at the bottom of the screen and places the cursor so that editing of the variable may take place. If there is no variable at the coordinates, RTES allows you to define one.

During the variable definition sub-mode, some more special keys are active:

<cr> Enter variable. If successful, leave the variable definition sub mode.

<Home> Leave variable definition sub mode without change.

<alt-B> Type out the word BINARY

<alt-L>	Type out the word LIST
<alt-N>	Type out the word NUMBER
<alt-R>	Type out the word REAL
<alt-T>	Type out the word TEXT
<alt-V>	Type out the word VGRAPH

CASE SIGNIFICANCE

Both upper and lower case characters are available. Variable names must be upper case.

If lower case characters are used in a command, as command word or argument, RTES generally interprets them as upper case.

There are some exceptions though. Since some lower case characters represent operators (a, i, o, n, d, s, t), RTES preserves the characters as typed in a line that may involve such operators. This includes the RULE command and the IF and LET task instructions. In such cases, point names must be typed in upper case only.

SEPARATORS

To allow proper interpretation of command lines, the various elements (arguments) must be separated by one or more spaces or commas.

In the case of an arithmetic or boolean expression, since the operators provide a form of punctuation, separators are not required but are acceptable.

CHAPTER FOUR
HOW TO DEVELOP A KNOWLEDGE BASE
(A TUTORIAL)

GETTING STARTED

The Real-Time Expert System, RTES, is contained in the executable file RTES.COM on a low density 5 1/4 inch or 3 1/2 inch diskette. If you have not yet installed RTES on your hard disk, place your RTES diskette in drive A:, make an empty directory on your hard disk the default, then type:

```
COPY A:*.*<cr>
RTES<cr>
```

When you exit from RTES the knowledge base files - OLEX.NAM, OLEX.PAG, OLEX.DAT, and OLEX.LNK - will be stored in your working directory.

EXAMPLE APPLICATION

Let's take the example of "Controlling a room heater" to demonstrate the process of developing a knowledge base.

The room heater should be turned on when the room temperature drops below a certain minimum temperature and remain on until the temperature rises above a maximum temperature. When the room temperature drops again the heater will not turn on again until the temperature drops below the specified minimum temperature.

DEFINING THE VARIABLES

For this example you will need three types of points or variables:

1. analog input for measuring the room temperature
2. digital output for turning the heater on and off
3. registers for specifying the minimum and maximum temperatures

Before we can define the rule that will turn the heater on or off, we must assign the point names.

To assign a name to the analog input for measuring the room temperature, type:

A<cr> to go to the first analog inputs page.

Then press: <Alt-E>

to go into the edit mode. The name area of the first point will be highlighted. To assign the name TEMP to the first analog input point, type:

TEMP<cr>

Now we need to define the type of conversion to be performed. Let's assume we have a linear device with a 0 to 5 volt output. To do this, we press:

<Up Arrow><Tab><Tab>

to highlight the conversion type area of the first analog input point. Because L is the default conversion method, we do not have to define the conversion method, i.e., we don't have to type anything. We do, though, need to define the zero and full scale values.

The zero value is the engineering unit value when the output from temperature sensing device is 0 volts. In this example, it would be the temperature reading when the input was 0 volts. Let's assume it to be -45 degrees. To define the zero value, you press:

<Tab>

to highlight the zero value area for the first analog point and then, type:

-45<cr>

to define the zero value.

Next we need to define the full scale value. In this example, it would be the temperature reading when the output from the temperature sensor is 5 volts. Let's assume the value to be 300 degrees. To define the full scale value you, press:

<Up Arrow><Tab>

to highlight the full scale value area for the first analog point and then, type:

300<cr>

to define the full scale value.

We have finished defining the room temperature sensor. To get out of edit mode and back to command mode, press:

<Home>

Next we can define the binary output point for turning the heater on and off. Type:

O<cr>

to go to the first binary outputs page.

Then press:

<Alt-E>

to go into the edit mode. The name area of the first point will be highlighted. To assign the name HEATER to the first binary output point, type:

HEATER<cr>

Press: <Home>

to get out of edit mode and back to command mode.

Next we can define registers for the minimum and maximum temperatures between which the heater will operate. Type:

R<cr>

to go to the first registers page.

Then press:

<Alt-E>

to go into the edit mode. The name area of the first point will be highlighted. To assign the name MIN to the first register and the name MAX to the second register type:

MIN<cr>
MAX<cr>

Before we get out of edit mode let us define the text to be used to display the status of the heater in our display page in our user interface. We'll define the following messages:

The heater is off
The heater has been turned on.

The first message will be displayed in red when the heater is off. The second message will be displayed in green when the heater is on. First we will assign the name H_MESSAGE so we can reference these messages. To do this, type:

H_MESSAGE<cr>

Then we need to define the text messages in the array. The first message in the array is the one associated with the name. Press:

<Up Arrow><Tab><Tab>

to highlight the message area associated with H_MESSAGE. Then type:

\4The heater is off<cr>
\2The heater has been turned on<cr>

We have now finished assigning the point names and defining the text messages that will be used. To get out of edit mode, press:

<Home>

BUILDING A DISPLAY PAGE

We can now define the display page where the operator can enter the minimum and maximum temperatures for control and where the status of the heater is displayed. We'll call the page H_CONTROL. To create the display page, type:

BUILD H_CONTROL<cr>

You are now in edit mode. Move the cursor along the top line of display page area using the

<Right Arrow> key. Notice that at the top of the page, the cursor column location is updated as you move the cursor along. When the cursor gets to column 30, type:

ROOM HEATER CONTROL

That will be the title of our display page. It forms part of the static information on the display.

Next we'll type the static message for the operator to enter the minimum temperature. Use the arrow keys to move the cursor to line 10 column 5. Type:

Minimum Temperature:

Now we'll define the variable MIN to be displayed as an integer to the right of the message and that the operator may change its value. Press:

<Alt-V>

At the bottom of the page a message will be displayed that tells the line and column where the variable is to be displayed. The cursor will be positioned at the end of this message. The define the variable MIN to be displayed as described above, type:

MIN/ NUMBER<cr>

The cursor will be returned to where it was before you pressed <Alt-V> and there will be a white block 5 columns wide in that location.

Next we'll type the static message for the operator to enter the maximum temperature. Use the arrow keys to move the cursor to line 11 column 5. Type:

Maximum Temperature:

Now we'll define the variable MAX to be displayed as an integer to the right of the message and that the operator may change its value. Press:

<Alt-V>

At the bottom of the page a message will be displayed that tells the line and column where the variable is to be displayed. The cursor will be positioned at the end of this message. To define the variable MAX to be displayed as described above, type:

MAX/ NUMBER<cr>

The cursor will be returned to where it was before you pressed <Alt-V> and there will be a white block 5 columns wide in that location.

Next we'll type the static message for the operator to adjust the room temperature for testing purposes. Use the arrow keys to move the cursor to line 15 column 10. Type:

Room Temperature:

Now we'll define the variable TEMP to be displayed as a real number to the right of the message and that the operator may change its value. Press:

<Alt-V>

At the bottom of the page a message will be displayed that tells the line and column where the variable is to be displayed. The cursor will be positioned at the end of this message. To define the variable TEMP to be displayed as described above, type:

TEMP/ <cr>

The cursor will be returned to where it was before you pressed <Alt-V> and there will be a white block 9 columns wide in that location.

Next we'll define the messages describing to the operator the current status of the heater. Use the arrow keys to move the cursor to line 20 column 20. Now we'll define the variable HEATER to be displayed as a string from the text array H_MESSAGE. Press:

<Alt-V>

At the bottom of the page a message will be displayed that tells the line and column where the variable is to be displayed. The cursor will be positioned at the end of this message. To define the variable TEMP to be displayed as described above, type:

HEATER LIST 2 H_MESSAGE 40<cr>

The cursor will be returned to where it was before you pressed <Alt-V> and there will be a white block 40 columns wide in that location.

We have now defined all the variables to be displayed. We can exit out of edit mode and back to command mode. To exit from the edit mode, press :

<Home>

When we do so, the white blocks are replaced by the current values of the variables being displayed.

DEFINING THE RULE

Next we have to define the rule describing when the heater will turn on. By default all other conditions define when the heater is off. To enter the rule for HEATER, type:

RULE HEATER=TEMP<MIN!HEATER&TEMP<MAX<cr>

TESTING AND DEBUGGING THE RULE

Now to test that everything is working okay, we enter values for the minimum and maximum temperatures.

Select the minimum temperature by pressing:

<Tab>

Then you need to type the number followed by <cr>. Let's set the minimum temperature to 20. Therefore, type:

20<cr>

Next select the maximum temperature by pressing:

<Tab>

Let's make the maximum temperature be 25. Therefore, type:

25<cr>

The message "The heater has been turned on" should be displayed in green.

Because we have not yet connected the analog input to the physical device we can also adjust the room temperature. Select the room temperature by pressing:

<Tab>

Then use <F8> to adjust the temperature up one degree at a time. Watch what happens to the heater. When the temperature reaches 25 the message "The heater is off" will be displayed in red where the previous message was displayed. Now use <F7> to adjust the temperature down one degree at a time. When the room temperature reaches 19 degrees, the heater will come on again.

Now you may want to review the control strategy of the heater. To review a rule you simply recall the rule by pressing <Alt-R> followed by the variable name and <cr>.

In this case, you will type

<Alt-R>HEATER<cr>

Once the rule is displayed you can modify it and re-enter it. A rule goes into effect as soon as you enter or re-enter it. Therefore, you can test any modifications immediately.

INTERFACING WITH THE REAL WORLD

Now that you have tested your control system, you can connect up the I/O Subsystem.

Let's assume you have an Optomux system set up in the following manner:

connected to COM1 at 9600 baud
address of analog board is 255
position of temperature sensor on board is 0
address of digital board is 0
position of output on board is 3
if communication is lost with the analog board, report the problem on register R0013
if communication is lost with the digital board, report the problem on register R0014

We use the above information to set up the configuration file. We can create the I/O configuration file PORT1.CNF by typing:

NOTE PORT1.CNF<cr>

The system blanks out a portion of the screen which you can use to write the contents of the file. Based on the above description of the Optomux system, type:

B=9600<cr>
A=1 0,1,255,0,12<cr>
O=1 0,1,0,3,13<cr>

To save PORT1.CNF to disk, type:

<Home>

LOADING THE OPTOMUX DRIVER

(a) exit out of RTES by pressing: <Ctrl-Home>
(b) load the driver by typing: OPTOMUX<cr>
(c) Then restart RTES by typing: RTES<cr>

When you exited from RTES, the knowledge base was stored automatically to disk. When you restarted RTES, the knowledge base is loaded into memory and the configuration file you created is also read and used.

DEFINING THE ALERTS

We should now define the alert registers, associated with the communication. Type:

R<cr>

to go to the first register page. Then press:

<Alt-E>

to get into edit mode. Then press:

<Up Arrow><Up Arrow><Up Arrow><Up Arrow>

to highlight the name area of the 13th register.

Let's call the alert register for the analog board ALARM_A and the alert register for the binary output board ALARM_O. To assign these names to the 13th and 14th registers type:

ALARM_A<cr>
ALARM_O<cr>

Next we define the alert state for these registers. RTES sets the alert registers defined in the configuration file to 1 when there is a communication problem. Therefore, the alert states for these two registers should be set to 1. To define the alert
we type:

<Up Arrow><Up Arrow><Tab>

to highlight the alert area of the 13th register. Then we type:

1<cr>
1<cr>

to define the alert states for the two registers.

Next we can define the messages associated with these alerts. We make the alert message for ALARM_A be "There is problem with the analog card" and the alert message for ALARM_O be "Trouble with the binary output card". To define the alert messages, press:

<Up Arrow><Up Arrow><Tab>

to highlight the message area of the 13th register. Then type:

There is problem with the analog card<cr>
Trouble with the binary output card<cr>

Now that we have finished defining the alerts and alert messages, we can go back to command mode by pressing:

<Home>

If you have not yet connected the communication lines, you would have had an alert come up as soon as you defined the alert states. The alert would have a blank message area. If you reset the alert by changing the value of the alert register to 0, you will see the alert come up again but this time it would have a message in the message area.

If you have connected the communication lines and the hardware is set up correctly, you will not get the alerts. To test that indeed an alert will occur, you can disconnect the communication line from the computer or turn the power off at one of your I/O

boards.

You may want to adjust the conversion factors of your analog input to reflect the characteristics of the temperature sensor you are using. The digital output should respond to the temperature changes.

CHAPTER FIVE - DATABASE

DATABASE

The RTES database is the collection of variables used by the system. The database includes the names assigned to the variables, the texts associated with those variables, the alert states, the analog conversion and linearization coefficients.

VARIABLES

Variables are symbols used to represent facts. The fact could be a numeric or a FALSE or TRUE status. The symbols used and the facts they represent are entirely at the discretion of the individual who establishes the relationships.

For example, the variable COST may be equal to a number, say 100.00. This could represent the fact that the cost of an item is $100.00.

The variable RAIN may be TRUE or FALSE, thus representing the fact that it is or it is not raining.

In the above cases, COST is a numeric variable, RAIN is a binary variable.

Generally, the symbols used would be an appropriate abbreviation, contraction or other easily recognizable form of the facts they represent. They constitute a mnemonic representation of the fact.

These symbols are called names. Each variable is assigned a name. Each variable name must be unique within the data base.

RTES allows the user to assign names to the variables. These names must start with a letter of the alphabet (A to Z), must contain only letters, numerals and the characters @ # $ % _ ? and must not be longer than ten characters.

RTES will not allow a variable name to be used more than once in the same data base. It is however possible to rename variables so that an already existing name can be freed for a new

use.

RTES takes care of its own cross referencing of variables. If the name of a variable is changed, all references to that variable - in the rules, the tasks and the user defined displays - automatically assume the new name.

Variables are also referred to as "points" in this manual, because they are normally associated with input points and output points found in the data acquisition and control system connected to RTES.

NUMERIC VARIABLES

Numeric variables may represent a real number with a sign (positive or negative), an integer portion and a decimal portion.

The range of magnitude of a numeric variable is from 2.4E-78 to 7.2E+75.

Numeric variables are listed in the "R" pages (for Register), the "A" pages (Analog Inputs) and the "Y" pages (Analog Outputs).

A numeric variable may be used as binary. RTES considers its integer value only and applies the rule:

```
+--------------------------+
|  even=FALSE, odd=TRUE    |
+--------------------------+
```

BINARY VARIABLES

Binary variables can only assume the values FALSE or TRUE. Arithmetically, these values correspond to 0 and 1 respectively.

RTES uses the OFF/ON terminology for binary outputs and the OPEN/CLOSED terminology for binary inputs.

Binary variables are listed in the "I" pages (Binary Inputs) and the "O" pages (Binary Outputs).

HOW TO DISPLAY THE VARIABLES

RTES internally identifies all variables with an index within each category (I,O,R,A,Y). The variables are kept in pages, each containing 16 variables.

You display these pages by entering the letter corresponding to the desired category followed by the page number. If the page number is omitted, RTES displays the first page of that category.

Once a page of variables is on the screen, the <Pg Dn> and <Pg Up> keys allow you to flip through the other pages within the same category.

Example:

R 3<cr> causes the 3rd page of "R" variables to be displayed. This shows variables R033 to R048.

Pressing <Pg Dn> once will then cause the display of page 4, showing variables R049 to R064.

The display shows the identifier number of each variable, the name currently assigned to it, the current value of the variable, as well as other characteristics discussed below.

With the exception of analog inputs and outputs, variables may also be accompanied by a text. The texts associated to variables may be used for several purposes as you will see in various sections of this manual. This text is limited to 40 characters.

BINARY INPUTS

The status of binary inputs is normally obtained automatically from the data acquisition subsystem attached to the Real Time Expert System.

It is however possible to change the status of a binary input (for simulations), using the point control commands. If communication with the data acquisition subsystem is actually taking place, the variable will return to the physical status read by the hardware.

The device driver(s) should not be installed during simulations, so that the data acquisition subsystem will not interfere.

Binary inputs cannot be evaluated arithmetically or logically. They cannot be assigned a rule.

They cannot be controlled by the clock scheduler.

Binary inputs may be assigned an alert state.

Binary inputs may be used as operands in rules and tasks.

The TRUE state of a binary input is represented by CLOSED, the FALSE state by OPEN.

BINARY OUTPUTS

Binary outputs are normally connected to the control part of the data acquisition subsystem. When communication is taking place with the data acquisition hardware, the status of a binary output is physically reflected by the status of the corresponding output point.

Binary outputs may be controlled by the point control commands, evaluated by a rule, or they may also be controlled by the clock scheduler and tasks.

Binary outputs may be assigned an alert state.

Binary outputs may be used as operands in rules and tasks.

The TRUE state of a binary output is represented by ON, the FALSE state by OFF.

ANALOG INPUTS

The values of analog inputs are normally established by the corresponding channels in the data acquisition subsystem using the evaluation method and parameters specified.

The data acquisition subsystem uses an A/D (analog to digital) converter to pass on to RTES a number representing the signal connected to the analog channel.

This number - the raw value - is from 0 to 4095. RTES uses the raw value and the evaluation method defined for that variable, to calculate the engineering value of the signal.

The evaluation method and parameters may be established by using the EDIT mode or database editing commands (see Database Editing).

The evaluation methods are :

(L)	Linear	(E=a+b*V)
(r)	Square root	(E=a+b*sqr(V))
(D)	100 Ohm RTD	
(J)	"J" Type Thermocouple	
(K)	"K" Type Thermocouple	
(R)	"R" Type Thermocouple	
(S)	"S" Type Thermocouple	
(T)	"T" Type Thermocouple	

The parameters, for L and R methods only, are set by deciding the "zero" reading and "full scale" reading in Engineering Units.

Example:

A load cell is connected to the channel corresponding to the analog input WEIGHT. The engineering unit is "lbs". The evaluation method is "L" (linear), the zero reading is set at 10 and the full scale reading is set at 350.

An A/D converter reading of 0 corresponds to 10 lbs, a reading of 4095 corresponds to 350 lbs.

When the A/D converter reads 500, RTES will calculate the value of WEIGHT, in engineering units, as:

500*(350-10)/4095+10=51.514 lbs.

The thermocouple conversion does not require any parameter, since the correspondence between mV and temperature is established by an industry standard formula.

Note that it is always possible to convert the A/D input to engineering units using a different method by defining a register and writing the rule that performs the conversion. The engineering unit would then be the value of the register.

The value of an analog input may be forced, using the point control commands. If communication is taking place with the data acquisition subsystem, then the value would return to the one set by the physical situation monitored.

The device driver(s) should not be installed during simulations, so that the data acquisition

subsystem will not interfere.

Analog inputs cannot be evaluated arithmetically or logically. They cannot be assigned a rule. They cannot be controlled by the clock scheduler.

Analog inputs cannot be assigned an alert state.

Analog inputs may be used as operands in rules and tasks.

ANALOG OUTPUTS

Analog output variables are normally connected to the control portion of the data acquisition subsystem. They are interfaced with the outside world using an D/A (digital to analog) converter that receives from RTES a number and converts it into a voltage applied to the equipment controlled.

A raw value of 0 to 4095, proportional to the analog output value is sent to the D/A converter. The proportion is calculated using the "zero" and "full scale" entered in the variable definition.

Analog outputs may be controlled using the point control commands and may be evaluated by rules. Their values, however, are not permitted to cause the raw value to go below 0 or beyond 4095. They cannot be controlled by the clock scheduler.

Analog outputs cannot be assigned an alert state.

Analog outputs may be used as operands in rules and tasks.

CONVERTERS USING OTHER THAN 12 BIT RESOLUTION

Some A/D and D/A converters use 14 or 16 bit resolution. The device drivers for such interfaces use the "R" registers instead of the "A" and "Y" to transfer to raw value to and from RTES. You may then define the rules to convert the input values to engineering units and to convert from engineering units to output values.

ON-SCREEN EDITING

RTES includes a full screen database editor. During an editing session, the cursor may be placed where the change or addition is to take place. New characters may then be typed over, or inserted and unwanted characters may be deleted. The <cr> key is then pressed to indicate that a new piece of information is to be entered.

RTES then checks the new entry for validity. If the new information is rejected, an error message is displayed. You may then correct your input, or abandon, in which case, the original information is re instated.

This editing method is available for all the information associated with variables. This includes names, text messages, alert states, conversion factors and parameters (for analog points).

You start an editing session with the EDIT command. You end the editing session when you press the <Home> key.

When a change is made, you may record it immediately on the disk with the SAVE command. Otherwise, then the change will be recorded on disk when the <Ctrl> and <Home> keys are pressed to terminate RTES.

It should be remembered that if the system shuts down abnormally, because of a power failure or a "re-boot", before the SAVE command is used, the changes will be lost. This also provides a way out if the changes made need to be discarded.

STARTING A DATABASE ON-SCREEN EDITING SESSION

Display a page containing the variable(s) to be edited, using the database display commands (O,R,etc.).

Enter the EDIT command. The format of this command is:

EDIT<cr> or <alt-E>

The top right hand of the screen will indicate "Editing POINT NAMES", and the cursor will be positioned at the first variable on the page.

SPECIAL KEYS

In addition to the line editing keys (arrow left, arrow right, backspace, delete, insert, end), the following keys have special functions during database editing.

The <Dn Arrow> and <Up Arrow> keys allow movement from one variable to the next.

The <Tab> key allows you to change the field being edited. This will be used to edit the alert states, text descriptors, etc. The top right hand of the screen indicates which field is being edited and the specific area of the screen where the editing will take place is highlighted. Whenever the <Tab> key is pressed, the next field is selected on the same variable.

The <cr> key causes the new entry to be checked and, if valid, entered in the database. When the <cr> key is pressed, its position in the field being edited is not relevant. The characters to the right of the cursor are retained as part of the entry. When the entry is accepted, the same field of the next variable is automatically selected for editing.

The <Pg Dn> and <Pg Up> keys allow you to change the page.

ENDING A DATABASE EDITING SESSION

The session may be ended at any time, and the system returned to command mode by pressing the <Home> key.

If the session was ended while still modifying an element, before the <cr> key was pressed, then the original information is retained for that element in lieu of the incomplete entry.

During a lengthy editing session, it is a good idea to save the changes on disk from time to time by using the SAVE command.

VALID ENTRIES

The table below summarizes acceptable forms for each field.

Field	Type	Range
Name	Alpha numeric	Up to 10 characters A-Z, 0-9, @ # $ % _. Must start with A-Z.
Alert State	Character	0=false, 1=true, E=either (record change of state), N=neither (cancel alert state)
(consult the chapter on alerts for more details)		
Text	Alpha numeric	Up to 40 characters
Eng. Unit	Alpha numeric	Up to 4 characters.
Conversion type	Character	L=linear, r=square root, D=100 Ohm RTD, J,K,R,S,T=thermocouple.
Zero	Real number	+/- .000001 to 9999999.
Full scale	Real number	+/- .000001 to 9999999.

Note : RTES does not accept scientific notation.

DATA BASE EDITING BY COMMANDS

You may also enter or change the elements associated with a variable without that variable being on the screen. This is particularly useful if you decide to build your data base using a batch command file that you create with a word processor.

You assign a name to a variable with the NAME command.

The format of this command is:

NAME Xn name<cr>

Where Xn is the variable identifier and "name" is the new name to be assigned to the variable.

Example:

NAME R5 TOTAL<cr>

This commands assigns the name TOTAL to the variable R0005.

You may also use : NAME . name<cr>

The "." means "the next variable". This allows you to name a series of consecutive variables by specifying the location of the first one only.

You assigned a text to a variable with the TEXT command.

The format of the command is:

TEXT name text<cr>

Where "name" is the name of the variable and "text" is a string of up to 40 characters.

Example:

TEXT TOTAL Total Consumption<cr>

Text can also be transferred from one variable to another with the TEXT command. The format of the text command to perform a transfer is:

TEXT destination @@source<cr>

where: destination is the name of the variable where the text will be placed
source is the name of the variable where the text is copied from

Example:

TEXT ABC @@ABCDEF<cr>

assigns the text associated with ABCDEF to ABC

If you select a variable on a I, O or R page, with the <Tab>, <Arrow Up> or <Arrow Dn> key, the TEXT command assigns any text after the word TEXT to the selected variable. To prevent this from occurring, make sure that no variable is selected by pressing the <Home> key before entering the TEXT command.

You may defined or removed an alert condition with the ALERT command. The format of the

command is :

ALERT name state<cr>

name is the name of the variable to be associated with the alert condition.

state is 0 for FALSE = alert condition
1 for TRUE = alert condition
E to log the change of state
N to remove a previously existing alert condition (consult the chapter on alerts for more details)

For analog variables you may define the engineering units, range and conversion type using the database editing commands.

You define the engineering unit with the EU command. The format of the command is:

EU name text<cr>

name is the name of the analog variable

text is the text representing the engineering units for the variable

You define the range with the RANGE command. The format of the command is:

RANGE name zero full<cr>

name is the name of the analog variable

zero is the "zero" value for the variable

full is the "full scale" value for the variable

You define the conversion type with the TYPE command. The format of the command is:

TYPE name x<cr>

name is the name of the analog input variable

x is the conversion method - or thermocouple type - to be used for the variable

CONTROLLING THE VARIABLES

BY COMMANDS

The START, STOP, SET and PULSE force a variable to take the desired value or state.

START, STOP and PULSE are normally used with binary variables. They may also be used with numeric variable when they assume a binary function.

The START command forces a variable to the TRUE state.

The STOP command forces a variable to the FALSE state.

The PULSE command forces a variable to the TRUE state for a defined period of time, then forces it to the FALSE state.

The SET command is used with numeric variables. It forces the variable to a defined value.

Example:

START MAN<cr> turns the variable MAN to TRUE.

STOP MAN<cr> turns the variable MAN to FALSE.

PULSE MAN 2.5<cr> turns the variable MAN to TRUE for 2.5 seconds, then turns it to FALSE.

If MAN was TRUE when the command was issued, it would simply be forced to FALSE 2.5 seconds later.

If the time is omitted, a default duration of 1 second is assumed.

SET COST 23.7<cr> Force the value of the variable COST to 23.7.

There is no default value with this command. Should the value be 0, it must be specified (e.g. SET COST 0<cr>).

BY FUNCTION KEYS

When a variable is displayed on a variable definition page or a 'debugged' rule, its value may be changed by selecting it and then using the increment/decrement keys.

This also applies to a variable specified on a customized display page, provided the specification included the / after the variable name. (See Customized Page Definition).

To select a variable, simply press the <Tab> or <Down arrow> key. The variable selected will be highlighted. To skip to the next variable, press the same key again. To go back to a previous variable, press the <Up arrow> key. The selection process wraps around so that the next variable after the last is the first and vice-versa.

If the selected variable has been specified as bar graph, highlighting is not possible. Instead, any text within the area occupied by the bar graph plus one more position beyond starts blinking. Remember that you must type the under the bar graph area you specify the bar graph variable.

Once the variable is selected the <F7> and <F8> keys may be used to adjust the value. If the variable is numeric, <F7> will decrease the value by 1 and <F8> will increase the value by one. Either key will invert the status if the variable is binary.

You may directly set the value of the selected variable. You simply type the desired value and press <cr> to transfer that value to the selected variable.

ON-LINE CALCULATOR FACILITY

When a variable is selected on a variable definition page, customized display page or a "debugged" rule, an expression enclosed in brackets may be entered on the command line. The value of the expression will be given to the variable. The expression may contain constants, operators and/or variable names. It uses the same syntax as the rules.

Example:

(2*X+3*Y)

THE "LET" COMMAND

You may assign a value to a variable using an boolean/arithmetic expression with the LET command. The format is :

LET name=expression

or: LET name i index = expression

where i is the index operator and index is a constant or a variable that defines an offset from the named variable. Spaces have been typed for clarity; they do not need to be present.

Example:

```
R0065 A     0.00
R0066 B     0.00
R0067 C     0.00
.
.
R0102 X    12.00
.
R0110 Y    10.00
.
R0202 F     2.00
```

LET — A=X+Y
yields A=22

LET — AiF=X+Y
yields C=22

The LET command is particularly useful when included in a batch file or used in connection with the EXEC facility (see the chapter on Algorithms for further information).

OVERRIDE

It should be noted that if the variable being controlled is otherwise evaluated by a rule, or a task, or is obtained from the data acquisition interface, the actual value will override the one entered.

THE "E" COLUMN

The column labelled E on the variable display pages, indicates which function is associated with each of the variables. The symbols used are:

blank	no function
A	ARC (arc cosine)
B	BITS (convert decimal to bits)
C	SHIFT (trigger for a shift register)
D	FDEC (convert bits to decimal)
E	EXT (external)
L	LOOP (loop control)
P	PID (PID loop control)
R	RULE
S	Time of Day Scheduler
X	EXEC (command list)

CHAPTER SIX - RULES

GENERAL DEFINITIONS

When we say that the desk is 5 foot long or that the temperature is 18 degrees, we state facts. Those facts assign a value to a variable by virtue of an observation or a measurement. The knowledge contained in those facts is DECLARATIVE.

On the other hand, when we say that the area of a rectangle is the product of the length by the width or that if the temperature is lower than 18 degrees we must turn the heating on, we apply RULES. Those rules represent PROCEDURAL knowledge. A collection of rules is called a KNOWLEDGE BASE.

RTES allows the representation of such rules as arithmetic and boolean (logical) expressions. Rules represent the "knowledge" of the system.

The general format of a rule is:

```
+---------------------+
| variable=expression |
+---------------------+
```

RTES contains an inference engine that detects changes in the variables and automatically triggers the evaluation of the rules that are affected by the variables that have changed. Whenever something changes in the expression on the right hand side, the inference engine applies the rule described by the expression and assigns the result to the variable on the left hand side.

Example no.1:

If WIDE, HIGH and AREA are three variables, then the rule

AREA=WIDE*HIGH

will represent that the system knows how to calculate -or deduce- the area of a rectangle given its width and its height. Whenever WIDE or HIGH changes, the system will change AREA accordingly.

The above rule is read "AREA equals WIDE times HIGH".

Example no.2:

If SUMMER, TEMP and ACUNIT are defined variables, then the rule

ACUNIT=SUMMER a TEMP>20

will represent that the system knows when the air conditioning unit should be on.

The above rule is read "ACUNIT is true if SUMMER is true and TEMP is greater than 20".

The "equals" or "is true" terminology is suggested by the format of the expression. If the expression yields a numeric result, then the left hand side of the rule equals the right hand side of the rule. If the expression yields a false or true result (binary), then the left side of the rule is true if the right side of the rule is true, false otherwise.

RTES does not put any restriction on what operations can be performed to yield a numeric or binary result. A binary can be considered numeric, in which case it represents the value 0 or 1. A numeric value can in turn be considered as binary, in which case RTES looks at the integer portion of the variable only
and applies the convention :

odd=true and even=false.

Rules are not applied in any specific order. They all apply concurrently. Since RTES has its knowledge base continuously on line, (it is constantly thinking) rules can be defined to translate a dynamic situation.

Example no.3:

Let us define the variables A and B and the rules :
A=n B d 2 (A is true if B is not true for 2 seconds)
B=A d 2 (B is true if A is true for 2 seconds)

The above will result in A going true for 2 seconds, then false for 2 seconds, and so on. B will turn momentarily true at the transitions.

The results deduced from rules, may apply to other rules.

Example no.4:

If, we add to example no.1 the numeric variables DEEP and VOLUME, then the rule:

VOLUME=AREA*DEEP

will cause VOLUME to be equal to the volume of the object defined by the 3 dimensions.

As long as a variable has not been assigned a new value, it assumes the current one.

Example no.5:

If we add to example no.3 the numeric variable C, then the rule C=C+A will cause C to increment every time A changes from 0 (false) to 1 (true).

HOW TO DEFINE OR EDIT A RULE

The key word RULE has a dual purpose.

If the key word is entered with a variable name followed by the equal (=) sign, RTES expects an expression after the equal sign. This expression will constitute a new rule.

The RULE key word may also be used to examine a rule. Simply enter RULE followed by the name of the rule. RTES displays the corresponding rule (if any). The cursor automatically positions itself at the beginning of the line. Pressing <cr> simply erases the command line, when you want to clear the command line.

In either case, once the rule is displayed, it can be modified using the editing keys, inserting and typing over, then re entered.

The name preceding the equal sign, identifies the rule. Whenever a rule is entered it automatically replaces any previously defined one with the same identification.

If, in the process of editing a rule, the name preceding the equal sign is changed, then the rule applies to the new identifier. This feature can be used to "clone" a rule from a similar existing one.

HOW TO REMOVE A RULE

The FREE command is used to delete an existing rule from the knowledge base. The format of the command is:

FREE name<cr> or <Alt-F>name<cr>

where name is the name of the variable currently evaluated by the rule to be removed.

STRUCTURE OF THE EXPRESSION

The expressions consist of operands (variables and constants) and operators arranged so that the operations are performed in a specific order on the operands to yield a result.

Most operators require two operands (+,*,etc.), some only require one operand (not, square root, etc.).

In a string of operations, the result of a previous operation becomes an operand for the next one. For example, 2*3*4 becomes 6*4, then 24.

In order to simplify the notation, certain operations are performed before others. It is said that some operators have precedence over others. For example, 2+3*4 becomes 2+12, then 14. In this case, the multiplication is performed before the addition although it is physically located after. The operator "*" has precedence over the operator "+".

Such rules of precedence are mostly governed by common usage. For instance, we all learned in Grammar School that arithmetic operations are done in the order: multiplication, division, addition, subtraction. (My Dear Aunt Sally).

RTES follows the same conventions but extends them to more operator types.

Whenever the operations need to be performed in a different order than the one dictated by the rules of precedence, brackets are used to create expressions within the expression. For example (2+3)*4 becomes (5)*4, then 20. The brackets define the addition as an independent expression within which the rules of precedence apply. The result of that expression is a new operand integrated into the overall calculation.

With RTES, brackets can be used and nested as required to produce the appropriate effect. In some situations, brackets could be redundant and may only help in clarifying the intent. RTES will maintain the brackets as entered. For example, it does not cause any harm to write 2+(3*4).

There are other considerations to be kept in mind when writing a rule. Certainly, the succession of operands and operators must be valid, the open and closed brackets must be paired, etc. These rules constitute proper syntax.

RTES will detect syntax errors and reject expressions containing such errors. An error message, as specific as it can possibly be, will be issued in such cases.

Table of allowed operators

Symbol	Operation	Key to produce the symbol
1. Arithmetic		
p	Power	"p"
e	Exponential	"e"
s	Sine	"s"
c	Cosine	"c"
l	Logarithm (base 10)	"l"
x	Logarithm (natural)	"x"
√	Square root	<Ctrl-S> (prints as sqr)
*	Multiplication	"*" or <Shift-8>
/	Division	"/"
+	Addition	"+" or <Shift-=>
-	Subtraction	"-" or <Hyphen>
2. Relational		
=	Logical Equal	"="
>	Greater Than	<Shift-dot>
<	Less Than	<Shift-comma>
≥	Greater Than Or Equal	<Ctrl-G> (prints as gte)
≤	Less Than Or Equal	<Ctrl-L> (prints as lte)
3. Logical		
' or n	Logical NOT	(apostrophe) or "n" X' is equivalent to nX
& or a	Logical AND	<Shift-7> or "a" X&Y is equivalent to XaY
! or o	Logical OR	<Shift-1> or "o"

X!Y is equivalent to XoY

4. Delays

d	Auto reset	"d"
r	Cumulative	"r"
t	Terminate	"t"

5. Miscellaneous

m	Largest positive integer	"m"
i	Index	"i"
(	Open bracket	"("
)	Close bracket	")"

PRECEDENCE OF THE OPERATIONS

RTES performs the operations in the decreasing order of precedence shown in the table below.

Index
Power and Exponential
Sine and Cosine
Logarithm (base 10 and natural)
Integer
Square root
Multiplication
Division
Addition
Subtraction
Relational
Logical Not
Delay
Logical And
Logical Or

HOW OPERATORS ARE USED IN EXPRESSIONS

Arithmetic operators

The power operator returns the value of the operand or expression immediately preceding the p raised to the power of the operand or expression immediately following the p. For example, the expression 2p3 yields the value 8 which is equal to 2 raised to the power 3. If you do not have a math coprocessor, the power must be an integer value.

The exponential operator returns the exponential of the operand or expression immediately following the e. For example, the expression e2 returns the value 7.389. The math coprocessor is required for the exponential operator to be in effect.

The square root sign is obtained on the screen by holding the <Ctrl> key down and typing "s" or "S". Since the character is not available in most printers, RTES replaces it by "sqr" at print time.

The square root operator returns the square root of the operand or expression immediately following the square root sign.

The sine operator returns the sine of the operand or expression immediately following the s. The cosine operator returns the cosine of the operand or expression immediately following the c. The units for the operand or expression immediately following the c and s operators is radians. The math coprocessor is required for both the sine and cosine operators to be in effect.

The logarithm operators return the logarithm (base 10 or natural) of the operand or expression immediately following the l and x operators. The math coprocessor is required for both logarithm operators to be in effect.

Other arithmetic operators (+,-,*,/) follow the rules of conventional algebra.

If a binary variable is used as operand in an arithmetic operation, it is considered as a numeric with the value 0 if FALSE or 1 if TRUE.

Note that the "-" is "subtract", not "negative". A negative constant must be expressed by subtracting it from 0, i.e. -3*X would be shown as (0-3)*X.

Note: If operators requiring the math coprocessor are used without the math coprocessor, RTES returns the value of the operand without performing any operation on it. For example, the expression e10 will return the value 10 rather than 22026.5.

Expressions that overflow (evaluate to numbers that are too large to be represented in the computer) return a value of 0.

Relational operators

Note that "greater than or equal" sign is obtained on the screen by typing <ctrl-G> and "less than or equal" is obtained with <ctrl-L>. Since these characters are not normally available on printers, RTES replaces them at print time by "gte" and "lte" respectively.

Relational operations compare algebraically the values of the two operands on either side of the

operator. The result of the operation is logical TRUE if the relation is verified and logical FALSE if the relation is not verified.

If a binary variable is used as operand in a relational operation, it is considered as a numeric with the value of 0 if FALSE or 1 if TRUE.

Logical operators

The AND operator is either "a" or &.
The OR operator is either "o" or !.
The NOT operator is "n" if it is preferred to place it before the operand, or ' (apostrophe) if it is preferred to place it after the operand.

Both notations can be used concurrently and are interchangeable.

The result of a logical operation is a logical TRUE or FALSE.

The result of the AND operation is TRUE if and only if, the operands on both sides of the operator are TRUE.

The result of the OR operation is TRUE if and only if, at least one of the operands on either side of the operator is TRUE.

The result of the NOT operation is TRUE if and only if, the corresponding operand is FALSE.

If an operand used in a logical operation is a numeric, then it is considered TRUE if its integer value is odd, FALSE if its integer value is even.

Delay operators

RTES allows two types of delays: Automatic reset and Cumulative.

In either case, the operator is immediately preceded by a binary operand called the trigger and immediately followed by a numeric operand (constant, variable or expression) called the duration. This duration is expressed in seconds.

The minimum duration of a delay is 0.1 second. If a delay duration evaluates to less than 0.1 second, it defaults to that value. The maximum duration of a delay is 3600 seconds (one hour). If a duration evaluates to more than 3600, it defaults to that value.

The resolution of the delay (clock tic) is 0.045 second.

An expression can only use one delay operation of either type.

Automatic reset time delay

The result of the "d" type delay operation is the logical TRUE if the trigger is TRUE and has been consistently TRUE for the specified duration. If the trigger changes state and never remains TRUE for the specified amount of time, the result of the "d" operation will always be FALSE.

The automatic reset time delay operates likes a pneumatic time delay relay. When the input is maintained true for a preset period of time, the output function operates. Whenever the input changes from false to true, the timing function starts all over again.

Example: X=Yd3.5

If Y is maintained true for 3.5 seconds, then X is true.
If Y is false, X is false.
If Y is true for any period less than 3.5 seconds, Y is false.

Cumulative timer

A "r" type delay is normally accompanied by a "t" operation. The "t" operator is immediately followed by a binary operand. The "t" operation returns the same value as that operand. The "r" operator is cancelled out whenever the result of the "t" operation is TRUE.

The result of the "r" type delay operation is the logical TRUE if the trigger has been TRUE for periods of time totalling the specified duration. Once the result of the "r" operation has been TRUE, it remains so until a "t" operation associated with the same rule, returns a TRUE result.

A "r" operation will always return FALSE if the associated "t" operation returns TRUE.

The cumulative delay function operates like a motorized timer with a clutch. As long as the clutch is engaged, the motor runs when the timed input is true. If the timed input turns false, the motor stops running but keeps the current elapsed time and goes on from there when the timed input turns true again. Regardless of how often the timed input switches off and on, the motor will

eventually accumulate a time equal to the preset value and the output will operate. Once the output has operated, the only way to deactivate it is to disengage the clutch, causing the accumulated time to go back to zero.

In RTES the 'clutch' action is represented by the 't' operator that should always be used in a rule that uses the 'r' delay.

Example: X=Yr2.3tZ

Provided Z is false, the timing function 'r' accumulates the time during which Y is true. When that time reaches 2.3 seconds - in one or in many shots - X becomes true and the timing function ends.

If Y turns false, while Z is false, then X does not change state and the time accumulated remains frozen.

If Z is true, then X is false and the accumulated time is reset to 0, regardless of the state of Y.

Variable indexing

During the evaluation process, each variable in the expression is replaced by its value or status. Indexing allows you to decide which variable to use at evaluation time. An indexed variable has of the form

NAMEiEXPR.

At evaluation time, the value of the expression EXPR is first established.

This expression could be a constant, variable name or any valid combination leading to an arithmetic result. The integer portion of the result is then added to the identification number of the variable NAME. The value of the new variable formed is then fetched. If EXPR happens to yield a negative result, then it is defaulted to 0.

This method allows the use of a series of consecutive variables as an array. It is however the responsibility of the user to ensure that the index produced by EXPR is valid.

Integer

The m operator returns the integer portion of a variable. For example, if Y is equal to 34.65

RULE X=mY returns 34 in X.

FAULT HANDLING

In order to retain the "on line" feature, RTES ignores errors generated by operations.

In general, underflow and overflow is ignored. The range of magnitude of a numeric variable is from 2.4E-78 to 7.2E+75. Should the result of an operation be beyond these limits, then the number returned is unpredictable.

Division by zero is allowed. It returns zero.

If the expression following the sqr sign is negative, the operation returns a zero.

If the result of an exponential expression is too large, the operation returns a zero.

A negative delay duration is equivalent to a 0.1 second delay.

"DEBUG" FEATURE

When the "DEBUG" feature is turned on, the above method of examining a rule also causes the screen to display all the variables used in the rule and their current value. Since the display is dynamic, you may then use the variable selection (Tab, Down Arrow, Up Arrow) and control (F7/F8) feature to analyze the operation of the rule.

The "DEBUG" feature is toggled by the command DEBUG. You switch it on by typing DEBUG<cr>. If it is on, you switch it off by typing DEBUG<cr>.

When the DEBUG feature is on, the word "DEBUG" is displayed in the top right hand corner of the screen in front of the time.

CASE SIGNIFICANCE

Since some lower case characters represent operators (a, i, o, n, d, s, t), variable names must be typed using upper case characters only.

SEPARATORS

Since the operators provide a form of punctuation, separators are not required but are acceptable. Separators are spaces and commas.

THE INFERENCE ENGINE

RTES uses a data driven - also known as forward chaining - inference engine. This confers to your application some characteristics that are quite different from conventional computer programs. Consider the following program (in BASIC or FORTRAN):

```
A=B+C
INPUT C
INPUT B
PRINT A
```

It will print 0, whatever value you entered for B and C, because when A is calculated, the values of B and C have not been established.

With RTES, if A is a rule (RULE A=B+C), then A will be evaluated when you enter the rule, evaluated again when you set the value of C, and evaluated again when you set the value of B. If A is displayed, it will always reflect the value that results from application of the rule. The order in which you have provided the information is not relevant. In other words, there is no 'program flow'.

Your application will typically contain several hundreds of rules. Only the ones that are affected by a change are triggered. The other rules are non-existent as far as the inference engine is concerned. Consider the following :

```
RULE A=B+C
RULE D=E+C
RULE C=X+Y
```

If B changes, A is triggered. If E changes, D is triggered. But if X or Y changes, then C is triggered then if the value of C has changed, A and D will be triggered.

Changes occur when a new value is received through the data acquisition interface. A change

may also occur because of an operator command, a task, a batch file that is being executed. In any instance, a change is extremely infrequent compared to speed at which the rules are evaluated. In general, the response time of your application will be dictated by the speed at which information is exchanged with the outside world. Should this not be the case, it would indicate that the computer that you are using is particularly slow for your application. You may consider using a faster computer and/or distributing your application over 2 or more networked computers (see NETWORK DRIVERS).

EXAMPLES AND TYPICAL APPLICATIONS

The following examples assume that the variables used have been previously defined using the data base editing facility of RTES.

1. Establishing that a variable is between two limits

The variable IN will be TRUE if VALUE is less than MAX and greater than MIN, otherwise it will be FALSE.

```
        IN=VALUE<MAXaVALUE>MIN

or      IN=VALUE<MAX&VALUE>MIN
```

2. Establishing that a variable is outside of two limits

The variable OUT will be TRUE if VALUE is greater than MAX or less than MIN, otherwise it will be FALSE.

```
        OUT=VALUE>MAXoVALUE<MIN
or      OUT=VALUE>MAX!VALUE<MIN
```

3. Dead band

The variable GO will become TRUE when VALUE is less than MIN and will remain TRUE until VALUE becomes greater than MAX, at which time GO becomes FALSE.

```
        GO=(GOoVALUE<MIN)an(VALUE>MAX)
or      GO=(GO!VALUE<MIN)&(VALUE>MAX)'
```

4. Linear ramp

VALUE will go from MIN to MAX in 100 successive increments at intervals of time equal to PERIOD, as long as INC is true. This application requires more than one rule.

The following two rules cause F to oscillate between TRUE and FALSE at interval PERIOD, as long as INC is TRUE and as long as VALUE has not reached MAX.

```
F=FX'd(PERIOD/2)&INC&(VALUE<MAX)
FX=Fd(PERIOD/2)
```

The following rule increments VALUE every time F changes from FALSE to TRUE.

```
VALUE=INC*(VALUE+F*(MAX-MIN)/100)
```

5. Start-Stop

MOTOR (a digital output) is controlled by 2 digital inputs ST and SP.

```
MOTOR=(MOTOR!ST)&SP'
```

6. Two-wire control (Flip flop)

LIGHT is switched on then off every time P is momentarily TRUE.

```
LIGHT=LIGHT'&P!LIGHT&P'
```

7. Average

Numeric variable AVR will contain the average of 3 variables T1, T2 and T3.

```
AVR=(T1+T2+T3)/3
```

8. Counting

Numeric variable to TOTAL will count how many times the binary variable ONE turned from FALSE to TRUE. Pulsing RST will zero TOTAL.

```
TOTAL=(TOTAL+ONE)*RST'
```

9. Closed loop Control (Regulation)

VALVE is an analog output controlling FLOW. SP is the desired flow and P is an adjustable constant (0<P<1). The following will adjust VALVE continuously to maintain FLOW at the set point SP.

```
VALVE=VALVE+P*(SP-FLOW)
```

The operation of the above loop may be simulated by entering a rule that calculates FLOW from VALVE, such as:

```
FLOW=VALVE
```

Note that the LOOP and PID functions allow you to implement a more complete loop control algorithms.

10. Multiple Choices

MONTH will contain the month number (1 to 12). The binary variable SPRING can be established by:

```
SPRING=MONTH=4!MONTH=5!MONTH=6
```

11. Time delay ON.

(Spaces are inserted for clarity only. Operands and operators do not need to be separated by spaces.)

The LIGHT comes on a certain period of time (DWELL) after SWITCH has been turned on.

```
LIGHT = SWITCH d DWELL
```

12. Time delay OFF.

The LIGHT goes off a certain period of time after the switch has been turned off.

```
LIGHT = (SWITCH' d DWELL)'
```

13. Cumulative delay ON.

The BELL rings when the CONVeyor has moved for a total of 10 seconds. One must press ACK to stop the BELL.

BELL = CONV r 10 t ACK

14. Cumulative delay OFF.

Provided the MOTOR is running, when the light BEAM has been interrupted (true) for a total of X seconds, stop the MOTOR.

MOTOR = (BEAM r X)' t MOTOR'

CHAPTER SEVEN - SPECIAL CONTROL ALGORITHMS

LOOP CONTROL

RTES allows the implementation of loop control using the Proportional, Integral and Derivative algorithm. Such control involves a control variable (CV) that regulates a measured process variable or feedback (FB) so that it is maintained at or near a given value - the set point (SP). This control is achieved by using operator defined parameters, determined experimentally during a process called "loop tuning" and that dictate how the CV will react at any given time considering the measured error (SP-FB).

The control variable is calculated as :

CV=CV+P*(E)+I*(E+Ep)+D*(E-Ep)

where P, I and D are parameters (typically between 0 and 1), and
E is the current error
Ep is the error measured the last time around.

The above calculation is performed whenever FB or SP changes, so that under fixed set point, CV is updated at the same frequency as the update of FB.

The loop may also be placed in "manual" mode, in which case CV is free to be set by the operator.

DEFINING A LOOP

To define a loop, you simply type in the command

LOOP name1 name2 name3 <cr>

where :name1 is name of the CV variable. This variable should be an analog output (Yxxx), but could also be a register (Rxxxx).

name2 is the name of the FB variable. This variable may be an analog input, analog

output or a register.

name3 is the name of the SP variable. This variable must be a register (Rxxxx).

RTES automatically assigns the next 5 register after name3 to become respectively the P, I, D, E and M variables.

These 5 registers may be assigned any names. RTES will use their current values to perform the algorithmic loop control calculation. The 4th variable (E) will be used to store the previous error. The fifth variable (M) is the AUTO/MANUAL switch. When M is FALSE, the loop is in auto mode. When M is TRUE, the loop is manual mode and is not controlled.

It is recommended to use a Yxxx variable as the CV, since this type of variables is bounded by the zero and full scale value. This prevents the loop control algorithm from wandering across a very wide range if the loop tuning parameters are not adequate or if the loop is not responding to the control for some mechanical reason. If you want the control variable to be a register, you should associate that register to an analog output, using a rule, and use the analog output as the CV.

VELOCITY TYPE PID LOOP CONTROL

The PID control algorithm used in the method described above, is known as the "position form". Some control systems require a more elaborate form called the "velocity form" PID. It uses the two previous values of the error, but allow a constant difference between feedback and set point if the integral term is not used. The calculation is :

CV=CV+P*(E-Ep)+I*(E)+D*(E-2*Ep+Epp)

To make RTES use the velocity form, define your loop using the key word 'PID' instead of 'LOOP' and reserve a 6th variable (after the Auto/Manual) to store the extra error reading.

OPTIONAL TRIGGER REGISTER

The PID and LOOP definition command line accept an optional (it does not have to be there) 4th argument: the name a R register that operates as follows.

When that register changes value, the loop or pid is evaluated, but the calculation is not in any way affected by the value of the register - only the current parameters of the loop are used in the

actual calculation.

This register is intended to be used in applications where it may be possible that the change in the feed back is not significant enough to return a different value through the A/D converter, but an error exists and you want to force the process to try and correct it.

You should be aware that if the feed back does not change, it may be because communication is not occurring or the hardware is otherwise defective. In this situation, you may very well cause the output to wind-up and overshoot once the communication is normally established.

DISPLAYING A LOOP

If only the CV name is used in the LOOP or PID command, the other parameters are displayed on the line. With the DEBUG option on, the variables involved will be dynamically displayed along with the P,I,D and M variables. You may then tune the loop by changing the parameters with the SET command or the <F7> and <F8> keys.

You may also group all the elements of the loop in a single dynamic display.

ADAPTIVE CONTROL APPLICATIONS

Since the loop tuning parameters, as well as the error(s) and the manual/auto switch are registers, their values may be used in rules, tasks and batch command files.

In particular, rules may be devised to continuously recalculate in real time the values of the loop coefficients, so that the loop is controlled optimally throughout the operating range of the controlled variable. This allows you to automatically adapt the loop tuning parameters to various operating conditions.

You might also want to use the Manual/Auto register to prevent loop operation beyond a certain range. Make sure, however, that you have a way to regain control of the loop, once the operation is within the range.

CASCADING LOOP CONTROL APPLICATIONS

In order to cascade loops, you simply have to express - using a rule or a task - the set-point of a loop as a function of the output of another loop.

CASCADING REGISTERS (SHIFT REGISTERS)

It is often useful to treat a series of variables as a block where upon occurrence of a certain event, the values are shifted from one variable to the next. For instance, it may be necessary to keep track of consecutive values of the same variable at different times. RTES provides this facility by allowing the definition of cascading registers.

The block of registers involved in the cascade is established by identifying the first register in the cascade - the one with the lowest identification number - and the quantity of registers in the cascade.

Another register will cause the shift to take place when its value changes from false to true.

DEFINING A REGISTER CASCADE

To define a cascade, you define the rule that controls the trigger register but you use the command word SHIFT in lieu of RULE.

RTES automatically reserves the two registers immediately following the trigger register to contain the number of the first register in the cascade and the quantity of registers in the cascade. If either register is null or negative, the cascade does not operate.

EXAMPLE

We want the readings of temperature (analog input) TEMP at 1 minute intervals to be displayed in a series of registers.

```
+-----------------------------------------------------------+
|      NAME  CURRENT  A E  MESSAGE                         |
|                                                                 |
|  -----                                                  |    |  R0012 T0      0.00
    TEMPERATURE NOW                          |
  R0013 T1      0.00      TEMP. 1 MINUTE AGO
  R0014 T2      0.00      TEMP. 2 MINUTES AGO
  R0015 T3      0.00      TEMP. 3 MINUTES AGO
  R0016 T4      0.00      TEMP. 4 MINUTES AGO
  R0017 T5      0.00      TEMP. 5 MINUTES AGO
```

The registers defining the cascade will be

```
+-------------------------------------------------------------+
|      NAME  CURRENT  A E  MESSAGE                           |
|                                                                |
|  -----                                                      |

|  R0100 TX      0.00       TRIGGER                          |
   R0101 FC     12.00    R   REFERS TO R0012 - 1ST REGISTER
   R0102 QC      6.00    R   REFERS TO LENGTH OF CASCADE
```

We now tell the system that TX is the trigger of a cascade

```
SHIFT TX=Md30
```

The following rule together with the previous one will cause TX to change from 0 to 1 (false to true) every minute.

```
RULE M=TX'd30
```

The following rule will cause T0 to contain the most recent value of TEMP

```
RULE T0=TEMP
```

At every transition of TX from false to true, the value of T5 will be replaced by that of T4, T4 by T3, etc. It is therefore possible to refer to the variables T1 to T5 as the previous values of TEMP, 1 minute ago, 2 minutes ago, etc. T0 to T5 may be displayed for 'dynamic trending' and used in calculations.

Note that the values of FC and QC may be changed at will or even established by some rules, so that the location and quantity of registers involved in a cascade can be defined by the process itself, as well as the variable associated with the first register of the cascade.

ARC COSINE

The ARC function returns the arc cosine angle in radians.

The ARC command has the following format:

```
ARC angle cos<cr>
```

angle register that will receive the angle in radians

cos register that will contains the cosine

Example: ARC ARCOS RATIO<cr>

A new angle will be placed in ARCOS each time the value of RATIO changes

COMMAND LISTS

The EXEC function allows you to treat the text associated with a list of registers as RTES commands to be executed under the control of a variable.

The EXEC command has the format:

EXEC name=expression<cr>

name name of the register which is used as index to a command list.

expression expression that determines the value of the index register.

Example: EXEC INDEX=1*(T<1)+2*(T>0&T<100)+3*(T>99)<cr>

```
+-----------------------------------------------------------+
|      NAME  CURRENT  A E  MESSAGE                          |
|                                                                  |
| -----                                                      |    | R0012 INDEX
0.00                                                 |
  R0013        0.00      SET SET_PT 125
  R0014        0.00      SET SET_PT 100
  R0015        0.00      SET SET_PT 50
```

When T is less than 1, INDEX will be 1 and therefore SET_PT will be set to 125. When T is between 1 and 99, INDEX will be 2 and therefore SET_PT will be set to 100. When T is greater than or equal to 100, INDEX will be 3 and therefore SET_PT will be set to 50.

The EXEC function is particulary useful in the design of intelligent operator interfaces. It is also used to initiate the execution of a task when a certain event occurs.

EXECUTING EXTERNAL CODE

The EXT function allows you to execute an external program.

This advanced feature is to be used in very special applications where the use of external code would substantially reduce the development time and complexity of your knowledge base. Evidently, you must write that particular external program in some programming language that allow its interface with RTES.

The EXT command has the following format:

EXT name=expression<cr>

name name of the register which is associated with the external code.

expression expression that determines the value of the register to be passed to the external code.

Example: EXT TRIGGER=TEMP<cr>

Each time the value of the associated register changes, RTES issues an INT 65H to DOS. CPU register AX contains the point identifier. CPU register DX contains the integer value of the variable. The RTES floating point register contains the floating point value of the variable. This allows several variables to be associated with the same external code.

The external code must be a memory resident program (TSR). The starting address of the external code should be placed at the interrupt 65H vector location (offset in 0000:0194 and CS in 0000:0196) prior to RTES being started up.

Your external code can exchange information with RTES with the use of software INT 67H. For detailed information on using these resources, see the Drivers chapter of the manual.

SAMPLE EXTERNAL CODE

The following program listing in 8086 Assembler illustrates the structure of an external program to be used in conjunction with the EXT command.

On startup this program checks to see if anything already exists at interrupt vector 65H. If the location contains data, the normal exit is taken. If the location is empty, the starting address of the external code to be executed is placed there. The program then informs the operator with the message "Sample external program". The program then uses the 'terminate and stay resident' exit to DOS.

The resident portion of the program executes each time the INT 65H instruction is issued. This occurs whenever the register associated with an EXT function is evaluated.

This sample program takes the value of register associated with the EXT function and places the integer portion of the value in the next 5 registers.

```
;;;;;;;     NAME  PROG ;;;;;;;;;;;;;;;;;;
;;;;;;;;SAMPLE PROGRAM - RECEIVES REGISTER n VALUE
;AND PUTS ITS INTEGER VALUE IN REGISTERS n+1 THRU n+5
GMEM SEGMENT        PUBLIC
          ASSUME          CS:GMEM,DS:GMEM
;OFSETS IN ROUTS (INT 67H FUNCTION NUMBERS)
SETPT   EQU     0       ;SET THE VALUE OF A VARIABLE
GTVAL  EQU     1       ;GET THE VALUE OF A VARIABLE
FLOGC EQU     2       ;CONVERT A 0 OR 1 TO FLOATING POINT
SETRV   EQU     3       ;SET THE RAW VALUE (FOR ANALOGS ONLY)
ASTFP   EQU     6       ;CONVERT AN ASCII STRING TO FLOATING POINT
FLOAT   EQU     7       ;CONVERT AN INTEGER TO FLOATING POINT
FPTAS   EQU     9       ;CONVERT A FLOATING POINT TO ASCII STRING
GTMSGEQU     10      ;GET THE TEXT MESSAGE OF A VARIABLE
SVMSG EQU     11      ;SAVE THE TEXT MESSAGE OF A VARIABLE
GETRV  EQU     12      ;GET THE RAW VALUE (FOR ANALOGS ONLY)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
BASE    DW      65H
;POINT ID MASKS
.Y                  DB      00100000B       ;'Y'
.A                  DB      01000000B       ;'A'
.I                  DB      10000000B       ;'I'
.O                  DB      10100000B       ;'O'
.R                  DB      11000000B       ;'R'
;
PROG  PROC  FAR
;VALUE IN DX & FPAC0 - POINT ID IN AX
                    STI                     ;RESTORE INTERRUPT CAPABILITIES
                    PUSH    AX
                    MOV     AX,DX
```

```
                MOV   BP,FLOAT
                CALL  RTES              ;INTEGER IN FLOATING POINT REGISTER
                POP   AX                ;GET POINT ID
                MOV   CX,5              ;COUNTER
NEXT:   INC     AX              ;NEXT REGISTER
                MOV   BP,SETPT
                CALL  RTES
                LOOP  NEXT
                IRET
PROG            ENDP
;TO EXECUTE A ROUTINE FROM RTES
RTES            PROC  NEAR
                PUSH  ES
                POP   DS
                INT   67H
                PUSH  CS
                POP   DS
                RET
RTES            ENDP
LAST_B  LABEL   BYTE
;;;;;;;;;DRIVER INSTALLATION ROUTINE;;;;;;;
INSTL   PROC    FAR
START:  SUB     AX,AX
                MOV   ES,AX             ;ABSOLUTE ADDRESS
                MOV   SI,CS:BASE                ;INTERRUPT NUMBER
                SHL   SI,1              ;*4
                SHL   SI,1
                CMP   WORD PTR ES:[SI+2],0  ;ANYTHING IN?
                JNE   NOINS                     ;YES. DO NOT INSTALL
                MOV   WORD PTR ES:[SI],OFFSET PROG          ;NEW INTERRUPT                  MOV
        ES:[SI+2],CS
                CALL  DMESG
                MOV   DX,OFFSET LAST_B
                ADD   DX,101h
                INT   27H
NOINS: MOV      AX,4C00H        ;NORMAL EXIT
                INT   21H
INSTL   ENDP
;ADVISE USER
DMESGPROC  NEAR
                PUSH  CS
                POP   DS
                LEA   DX,MESG
                MOV   AH,09H
                INT   21H
                RET
DMESGENDP
```

```
mesg   db      'Sample external program',10,13,'$'
GMEM           ENDS
               END     START
```

REMOVING A FUNCTION FROM THE KNOWLEDGE BASE

The FREE command eliminates a previously defined function (LOOP, PID, ARC, EXEC, EXT) from the knowledge base.

The format of the command is:

FREE name<cr> or <Alt-F>name<cr>

name name of the register associated with the function.

TIME OF DAY SYNCHRONIZATION

This feature allows you to set the clock using the RTES registers. It may be used to synchronize the time in all RTES nodes of a network.

To use this function, RTES.cnf must contain:

K=n

where n is the index of the first of 4 RTES registers that are used as follows:

the command register
hour
minute
second

When you set the command register to 1 (true), the integer positive contents of the next 3 registers are used to set the time. The command register is automatically returned to 0 after the operation.

BIT HANDLING FACILITIES. (FDEC & BITS)

The FDEC command produces a decimal number from a series of bits

contained in a register array. The syntax is:

FDEC destination array n[CR]

destination is the name of the R register to contain the resulting decimal number
array is the name of the first I,O or R register to be included in the calculation
N is the number of register to include in the array (1 to 16)

Example:

```
FDEC X STATUS 5

R0080 X         31.00    D
.
.
R0095 STATUS     1.00
R0096            1.00
R0097            3.00
R0098            1.00
R0099            1.00
```

The BITS command breaks up a decimal number into a series of bits. The syntax is:

BITS array source n[cr]

array is name of the first of the n I,O or R registers that will be set or cleared
source is the R register that contains the decimal number to be decomposed
N is the number of registers in the array

Example:

```
BITS X1 VALUE 3

I0016 X1  CLOSED    B
I0018 X2  OPEN
I0019 X3  CLOSED


R0100 VALUE    5.00
```

Note that the bit value of a register is derived from its positive integer value. Odd is 1 and Even is 0. A negative value is 0.

CHAPTER EIGHT - TASKS

OVERVIEW

A task is a list of instructions.

An instruction generally consists of a number called the label, followed by a key word called the command, then followed by arguments such as a label, variable name, value, expression, etc.

Example.

```
1 SET UNIT 23.5          (set the value of variable UNIT at 23.5)
2 START FIRST            (force the value of FIRST to true)
3 WAIT 2.5               (suspend execution for 2.5 seconds)
```

When the task "runs", the instructions are executed.

Normally, the execution process steps through the instructions in the sequential order of their labels. There are however, certain instructions that may modify that sequence by telling the execution process which instruction to execute next.

Such "branching" may be unconditional or conditional. They provide the facilities to produce loops and decisions.

Example.

```
4 IF [TOTAL>0] 10   (conditional branch)
5 GOTO 1                 (unconditional branch)
```

RTES allows the definition and simultaneous execution of several such tasks. Each task is identified by name and may be individually created, modified, initiated, suspended or terminated while the rest of the system and the other tasks continue operating.

In fact RTES even allows you to modify a task while it is running. You should use your own judgement in this situation, especially if the modifications affect the order in which the instructions are executed.

The instructions that a task may execute fully interface with the knowledge base. It may perform time dependent functions to support the knowledge base, or simply define values and set conditions.

Tasks may also be used independently from the knowledge base to provide a simple yet powerful form of programming language - a fingertip alternative to BASIC for most little programs that one may need to write and run.

HOW TO CREATE A TASK

The command TASK allows you to create a new task by name. The format of the command is

TASK name<cr>

Example: TASK RAMP<cr>

The name of the task may consist of letters, numerals and @ # $ % _ ? and may have up to ten characters. Each task must have a different name.

Provided the name you picked is acceptable, RTES creates the task and opens it for editing. The work area of the screen is cleared and the cursor is positioned to accept the first instruction.

You may now type your new task. Whenever you have completed typing an instruction, you must press <cr> to tell RTES to check it. You may use the editing keys to correct typing mistakes. If you want to move from one instruction to another, you may do so by using the <up arrow> and <down arrow> keys.

If you modify an instruction, you must press <cr> to have it re checked, otherwise, the modification is not implemented.

If you press <cr> on a line that contains only a label, the corresponding instruction is deleted (if it existed).

The instructions you type do not necessarily have to be in the proper sequence. RTES will arrange them in order of increasing labels. Labels may be skipped to allow for easier insertions.

You may at any time leave the task editing mode, by pressing <Home>. You must then use the command OPEN to get back to it.

HOW TO VIEW OR MODIFY AN EXISTING TASK

The OPEN command allows you to open a task for editing. The format of the command is:

OPEN name<cr>

Example: OPEN RAMP<cr>
The screen will then display the first 16 instructions of the task and the cursor will be positioned at the start of the first instruction.

You may then use the arrow keys to move the cursor around to the location where you want to make the change. If the task contains more than 16 instructions, pushing the cursor down causes the other instructions to appear.

Whenever you change an instruction, you must press <cr>, otherwise RTES simply ignores the change.

To leave the task edit mode, you press <Home>.

TASK DIRECTIVES

These are the directives that may be used in a task.

END Terminate the task.
GOTO Unconditional branch.
HALT Suspend execution of the task.
IF Branch if condition is true.
LET Assign a calculated value to a variable.
ONEND Define a label for task termination.
REM Introduce a line of text as a comment.
SET Assign a value to a numeric variable.
START Force a binary variable to TRUE.
STOP Force a binary value to FALSE
TRACE Define an RTES register to track the task execution.
WAIT Suspend execution for a period of time.
X Execute the rest of the line as an operator command

To reduce the amount of typing, some command words are generated by function keys.

F1, F2 and F3 generate START, STOP and SET, respectively. The other command words are obtained by holding the <alt> key and pressing the first letter of the word.

The following task directives are specifically for file handling.

APP Set record position to end of file.
ERROR Define a RTES register to report file handling errors.
GPOS Get current record position.
OPEN Define a set of files - floating point ASCII format.
BOPEN Define a set of files - 32 bit floating point binary format.
IOPEN Define a set of files - 16 bit integer format.
POS Set record position.
READ Read a record.
WRITE Write a record

SYNTAX

An instruction must start with an integer label that identifies the position of the instruction in the sequence. The label and therefore the number of instructions that a task may contain, must not exceed the limit shown in Appendix B (Configuration).

The next element must be a command word. Then come the optional arguments.

The elements of the instruction must be separated by spaces or commas.

The formats of the instructions are :

label END<cr>
label HALT<cr>
label GOTO label<cr>
label IF [logical_expression] label<cr>
label LET variable=expression<cr>
label REM text<cr>
label ONEND label<cr>
label SET numeric_variable value<cr>
label START binary_variable<cr>
label STOP binary_variable<cr>
label TRACE register_name<cr>

label WAIT time<cr>
'time' may be a numeric or the name of a variable. Unit is seconds.
label X command<cr>

The logical expressions and expressions used in the IF and LET instructions follow the same syntax as those in the rules.

Example:

The following task will add the value of STEP to the value of LEVEL. The operation will be repeated 10 times, at 3 second intervals. The numeric variables STEP, LEVEL and C are defined.

```
1 SET C 10
2 LET LEVEL=LEVEL+STEP
3 LET C=C-1
4 IF [C>0] 6
5 END
6 WAIT 3
7 GOTO 2
```

INSERTING AND DELETING LINES

During task editing you may insert and delete lines with the <Ctl-I> and the <Ctl-D> keys.

<Ctl-I> clears the line where the cursor is located, moves the current line to the next line down and increments all the line number from the cursor down. Any instruction that references a line number (IF, GOTO and ERROR) is adjusted accordingly. Any lines above the cursor are not redisplayed but they will show the correct line reference - if any - when redisplayed. Any reference to line 132 remains unchanged. This function is disabled if line 132 is used in the task.

<Ctl-D> decrements the line numbers from the line where the cursor is located down t the end and adjusts the line references for the IF, GOTO and ERROR instructions. This function is disabled if the line number immediately preceding the one where the cursor is located is used. In other words, <Ctl-D> does not erase a line, it simply closes a gap in the line numbering if that gap exists. To delete a line, you must still unter the line number witthout anything after it.

INDEXING WITH THE "LET" TASK DIRECTIVE.

The LET task directive allows the use of the indexing operator on the left hand side of the equate. It must be followed by a variable name to be used as index. The result of the expression on the right hand side of the equate is assigned to the variable that the left hand side points to, using the base name and the value of the index.

Example: LET XiJ=A+B

If X is R0015 and J is equal to 10, the value A+B is assigned to R0025. If J is equal to 0, then A+B is assigned to R0015.

FILE HANDLING

Using tasks, you may read and write disk based, record oriented files. The following task instructions allow you to define the files, alter the read/write position to the beginning of any record and exchange data between the RTES registers and the disk file.

The format of the data depends on the type of "open" statement used. When using the OPEN statement, the data in the disk file is kept in text format, each number occupies 11 positions including two decimals. The numbers are separated by commas. Each record is terminated by a carriage return/line feed. Files created by this process are compatible with most spread sheets and commercial data base software.

When using the BOPEN statement, the data in the disk file is kept in 32 bit binary format. There are no separators between the numbers. Access time for files opened with the BOPEN statement is less than that for the files created by the OPEN statement.

When using the IOPEN statement, the data in the disk file is kept in 16 bit positive integer format. There are no separators between the numbers. File access for these files is faster than those created with the OPEN statement.

In order to use a file, you must first define it with one of the "open" statements - OPEN, BOPEN or IOPEN. The format is

label open n reg_name l<cr>

where open is the open statement for the type of file to be used. OPEN for

ASCII floating point files. BOPEN for 32 bit binary files. IOPEN for 16 bit integer files.

n is the file number (1 to 16)

reg_name is a RTES register that points to a family of file names. You write the actual file name(s) in the text area pointed by the register, one file name per line. The integer value of the pointing register reg_name is used as an index to select which file name is to be used at the time a file handling statement is executed.

l is the record length. It represents the quantity of consecutive registers that will be read or written as a record. Maximum record length is 16 if in ASCII format, 256 if in binary or integer format.

You may define an error reporting register with the statement:

label ERROR reg_name<cr>

The specified register reg_name will be associated with the specific task that executed the statement and will contain an error code at the end of each file handling operation.

An error code 0 means that the operation was successful. An error code of 6 means that a READ was attempted past the end of the file. There are no other error codes defined at present.

Once an "open" statement has been executed, file n is active and you may manipulate its records. Note that file n is actually a family of files defined by the "open" statement. Which of the files is physically accessed will depend on the value of the register used in the "open" statement, at the time the operation takes place. When the read/write position is changed, it is changed for all the files in the family. Also, all the files in the same family automatically have the same record length.

label POS n reg_name<cr>

sets the read/write position of file n at the beginning of the record number contained in reg_name. The first record is 0, so that if reg_name has a value of 0, the read/write position is at the beginning of the file.

label GPOS n reg_name<cr>

returns in reg_name the current read/write position of file n.

label APP n<cr>

places the read/write position at the end of file n - past the last record.

label WRITE n reg_name<cr>

takes the values of the l consecutive registers (l is defined by the OPEN statement), starting at reg_name, and writes them to file n, at the current read/write position. After completion of the write operation, the read/write position is automatically at the beginning of the next record. (You may keep track of it by executing a GPOS after executing the WRITE).

label READ n reg_name<cr>

reads the values of the l consecutive registers (l is defined by the OPEN statement), starting at reg_name, from file n, at the current read/write position. After completion of the read operation, the read/write position is automatically at the beginning of the next record. (You may keep track of it by executing a GPOS after executing the WRITE).

NOTES:

A series of consecutive read/write operations automatically access file n sequentially, since the read/write position is incremented after each operation.

There is no 'close' statement. RTES closes the file - in the DOS sense - after every file access. If you re-use a file number n in another "open" statement, within the same task or within a different task, the new file specification takes effect and replaces the previous one.

A family of files defined by an "open" statement in a task, may be used by other tasks. The same files may be referenced by several "open" statements of the same type, under different file numbers. For each family of files, a separate read/write position is maintained.

An example of how to use these file handling capabilities is described in the application notes at the end of this manual.

HOW TO CONTROL THE EXECUTION OF A TASK

The RUN command allows you to start execution of an inactive task or to resume execution of a suspended task.

The format of the command is:

RUN name<cr>

Example: RUN RAMP<cr>

Note that <F5> generates the command word RUN.

The initiated task will terminate spontaneously if it executes an END instruction, or if it has no more instructions to execute.

When a task executes a HALT instruction, it stops executing. It remains suspended until you reactivate it by issuing another RUN command. When re-activated, execution resumes with the instruction following the HALT.

You may suspend a task while it is executing with the HALT command. The format of this command is:

HALT name<cr>

Example: HALT RAMP<cr>

Note that <F6> generates the command word HALT.

When a task receives a HALT command, it suspends itself after completing the instruction currently being executed. When re-activated by a RUN command, it resumes execution from where it left off.

If you suspend a task while it is executing a WAIT instruction, it will continue its countdown which is part of the current instruction.

You may terminate execution of a task at any time with the END command. The format of the command is:

END name<cr>

Example: END RAMP<cr>

This command will terminate the task immediately after the completion of the current instruction. If the task was suspended or waiting, it is terminated as soon as the END command is issued.

ONEND STATEMENT

The ONEND statement in a task introduces a line number that will gain control whenever a task is terminated for any reason. As control is passed to that line, the ONEND is cancelled so that the task would actually terminate the next time it executes an END or is terminated externally. If the ONEND statement defines a line number that does not exist, control is passed to the next consecutive line number that exists.

If a task is suspended at the time it is terminated, the task re-activates as it starts executing at the ONEND statement.

This feature is useful in cases where a task must perform some 'cleanup' before terminating. (orderly shutdown).

Once the ONEND line has gained control, an ONEND statement may again be executed so that termination of the task follows the same mechanism. This could be used to make a task re-synchronize (re-start at a specific line number) by terminating it.

Example :

```
        1 ONEND 100
-----
        100 STOP PUMP
```

ONE STEP AT A TIME TASK EXECUTION

You may make a task step through one line at a time with the STEP command. When you type

```
        STEP task_name<cr>
```

the task executes one line and suspends itself, waiting for the next STEP command or a RUN command. If the task had previously executed a TRACE command, the register defined by that command shows the line number to be executed next.

MISSING INSTRUCTIONS

If a task is directed to execute an instruction by IF or GOTO and there is no instruction with the label specified, it simply executes the next existing instruction after the missing one. If none is found, the task terminates.

Example:

```
1 IF [GO] 5
3 GOTO 1
```

The above task will loop continuously until the variable GO is TRUE. At that time, it will attempt to execute the instruction with the label 5. Since the last label used is 3, the task will terminate. If there was an instruction with label greater than 5, it would gain control.

This is particularity important to consider when you are editing a task while it is still executing. You may, during the editing, remove certain instructions or direct the task to an area where there is nothing more to do.

In the latter case, the task will simply terminate. In the former situation, the task may execute certain statements that it would not normally, had the modification been complete. In such situations, it is recommended to terminate the task, make the changes, then re-activate it.

Of course, there is no need for such precautions if the changes are not of a nature that will detrimentally alter the flow of execution. It is perfectly feasible to change a constant, for instance, in a task while the task is running.

MONITORING TASK STATUS

If the command word TASK is used without specifying a task name, RTES provides a dynamic display of the task activities. This shows all the defined tasks and their current status (inactive, active, waiting or suspended).

This command is also issued by pressing <alt-T>.

The command word TS also performs the same function.

HOW TO CHANGE THE NAME OF A TASK

The REN (rename) command allows you to assign a new name to an existing task.

The format of the command is:

REN old_name new_name<cr>

Example: REN RAMP STEPS<cr>

The new name must comply with the rules for a task name.

HOW TO REMOVE A TASK FROM THE SYSTEM

The ERASE command allows you to remove all trace of a task, as if it had never been created. The format of the command is:

ERASE name<cr>

Example: ERASE STEPS<cr>

Caution must be exercised with this command since there is no elegant way to recover from it. If the lost task happens to have been saved on disk, you could stop your computer with a soft boot or a power off and restart with your old data base.

If the task was executing when you erase it, it terminates at the end of the current instruction, then disappears.

NOTE REGARDING THE USE OF THE "X" COMMAND

Any text used in this command will be used by the Real-Time Expert System just as if it had been typed on the command line. Caution should be exercised because, provided the syntax is correct, the command will be executed. You may, for instance, modify a rule or cancel it; enter or remove an event in the scheduler; display a user page; or run or end another task. Since several tasks may be executing simultaneously, the results can be quite unpredictable. For instance, the displayed page may change constantly or a rule created by one task may be removed by another.

AUTOEXEC

On startup, the Real-Time Expert System will look for a task called AUTOEXEC and, if found, run it. You may include in such a task any commands that you wish to execute automatically on system startup, such as set the value of certain parameters, display a page of your choice, initiate execution of other tasks, etc.

USING TASKS AND RULES

The multitasking environment of RTES runs the active tasks and the inference engine of the rules totally asynchronously. When a task modifies the value of a RTES variable, the corresponding rules are triggered, but are not necessarily evaluated before the next statement of the task is executed.

If the operation of a task is dependant on the value of a variable that itself depends on the result of a calculation, it is advisable to consider using LET statements in lieu of rules. In this way, the evaluation of the variables is sure to take place before there values are examined by the task.

TASK SYNCHRONIZATION

In some situations where a task depends on what another task is doing, you could use a register as a 'mail box' allowing a task to inform another of its present status.

Example:

```
.                               ;Z is a function of X
09 LET X=Y                  ;This is the new value of X
10 STOP DONE                    ;Z is calculated by another task
11 IF [DONE'] 11                ; that will turn DONE on after calculating Z
12 IF [Z>MAX] ..                ;The new value of Z is available.
.                               ;Proceed.
```

In particular, if your task uses printing or batch files, you should make sure that those activities are completed before proceeding to the next step; otherwise, you may end up with incomplete execution of the print or batch activity. The monitoring of these activities is easily done with the reserved registers associated with these functions.

Example:

```
.
.
10 X PRINT HEADER
11 IF [PRINT$] 11
.
.
20 X GET SETUP
21 IF [READING$] 21
.
.
```

We recommand that you group all printing activities or all batch file execution activities in a single task that handles all the requests of the same category. This makes it easier to monitor these activities and prevent them from overlapping.

USING TASKS EFFICIENTLY

The RTES tasks run in parallel with the inference engine. As long as a task is active, it keeps going round and round - wether it is doing something useful or not. In this sense, it resembles any conventional program that is not 'event driven'. Therfore, task execution is substentially slower than the application of rules. Furthermore, the more tasks are active, the slower they get. In extreme situations, they may even impact on the speed of the inference engine.

Tasks are not an alternative to rules, although they appear attractive in that sense, particularly to programming language adepts. Using tasks instead of rules, would make your application slow and somewhat defeat the purpose of using RTES.

Tasks are designed to handle situations that clearly dictate their use:

- involved in file handling, printing, batch execution, pixel graphics.
- using an iterative process that requires program flow
- designed to supplement/enhance the operator interface

Note that the above activities are not time critical. In general terms, before doing anything in a task, ask yourself the question : "can it be done with rules?".

Use the EXEC function to initiate a task when necessary instead of letting it continuously check for a certain condition to occur

Example :

The user turning on	R0010 REP_TASK 0.00 X R0011 0.00 RUN REPORT R0012 REP_1 0.00 R0013 REP_2 0.00 .	requests a report by REP_1 or REP_2.
The EXEC command	EXEC REP_TASK=REP_1!REP_2	function executes the RUN REPORT.
The task which report is the Note that the that initiated it the next	EDITING REPORT 001 IF [REP_1] 20 002 IF [REP_2] 40 003 END . 020 STOP REP_1 . 040 STOP REP_2 .	REPORT establishes requested and goes to corresponfing line. task cancels the flag to make it ready for request.

HOW TO PRINT A TASK LISTING

To obtain a hard copy of a task, you type the command:

PT task_name<cr>

For example : PT PURGE<cr>

DISK RESIDENT TASKS

You may write a task using your own word processor and save it as a DOS text file. The first line of that file must contain only the name of the task.

You can import such files into RTES with the FTASK command.

For instance if the DOS text file "T1.TXT" contains:

```
+----------------------------+
|       AVERAGE                |
|       10 LET AV=(A1+A2+A3)/3     |
|       12 WAIT 20                 |
|       13 GOTO 10                   |
+----------------------------+
```

When you execute the RTES command:

FTASK T1.TXT<cr>

RTES will create a task called AVERAGE containing the 3 above lines.

The FTASK command should not be used when RTES is executing time critical activities, since it puts the multitasking environment on hold while the file is read and interpreted.

REPORT GENERATOR WITH TASKS

Two new directives have been added to the RTES tasks : DOC and PRINT. Although they basically perform the same functions as the RTES commands of the same name, the way they operate is quite different.

The similarities are:

DOC filename opens/creates a destination file to append text to.

```
DOC          opens/creates a destination file pointed to by
@filename    the text array variable "filename"
DOC          closes the destination file
PRINT        sends the text in filename' (source) to the
filename     previously opened file via an interpreter that
             resolves the @@variable@ and the
             [[condition]text]
PRINT        uses the text in the file pointed to by the text
@filename    array variable "filename"
```

The differences are :

- (1) Each task owns its DOC and PRINT directives and several tasks may execute them in parallel, each task with its own file.
- (2) The PRINT directive will retain control as long as there is something to print in the referenced file. There is no need to check if the text output is over. The line following the PRINT directive executes only after the PRINT has finished its execution.
- (3) A PRINT without a preceding valid DOC, fails - it does not do anything. If you actually want to use the printer, you should specify PRN as the destination file.
- (4) A DOC that specifies a new destination file within the same task closes the previous destination file before opening the new one.
- (5) If a task terminates, it closes the current destination file if any.

Example:

```
10 LET FILE=DAY$
11 DOC @FILE
12 PRINT STATS.RPT
13 END


R0020 FILE   0.00
R0021        0.00     SUNDAY.WRP
R0022        0.00     MONDAY.WRP
R0023        0.00     TUESDAY.WRP
.
.
```

"STATS.RPT" will contain the format for a (multi page) report

containing statistics to be kept for each day of the week.

TEXT HANDLING DIRECTIVES: MOVE, ASC and BIN

The format of these commands is
MOVE [source] [destination] qty
ASC [source] [destination] qty
BIN [source] [destination] qty

Source and destination may be a register or an indexed register. qty is the quantity of registers to use, starting at the named location(s).

The MOVE directive allows to copy a series of consecutive registers to another location.

003 MOVE [RECIPE] [NEW] 12
or
003 MOVE [BLOCKiCURRENT] [DISPLAYiLINE] 12

The ASC directive converts the values of a set of consecutive registers starting at the named source, into a string placed as a text for named destination register.

The BIN directive converts the text at the named source register into a set of values placed at the qty set of registers starting at the named destination register.

CHAPTER NINE - DISPLAY GENERATOR

OVERVIEW

The display generator allows you to prepare your own customized displays containing text and variables. The displays are dynamic. When a variable is on a display, it is automatically updated to reflect its current status or value.

A variable may be shown as a decimal number, an integer, a character string or a bar graph. The status of a binary variable may also be represented by a word selected from a pre defined list. When a number is used, the number of digits (field width) may also be selected. A variable may also be interpreted as a color change.

It is also possible to a make variable accessible by the operator, in which case, it may be selected and changed by using the function keys.

Displays may also be enhanced by using the "draw mode". This feature allows the drawing of tables, boxed titles, etc. to make the display more readable.

Customized displays may be used to present information to the user and/or to allow data entry.

You may use the display generator while RTES is operating. This means that you can modify or create displays as the need arises, while your knowledge base is on line and while other tasks are running.

HOW TO CREATE A NEW DISPLAY

To create a new display, you must use the BUILD command word followed by the name you want to give to the new page. This name may use any combination of letters, @,#,$,%,_ and numerals and may contain up to 10 characters.

Example: BUILD FORM1<cr>

Will create a new page called "FORM1".

Each display must be assigned a unique name. RTES will not allow a display to be created with a name that has already been used for a display. That name, however, may also identify other entities such as variables and tasks.

For example, the display FORM1 may be created if there is a register called FORM1 in the current database, but may not be created if there already is a display called FORM1.

When you enter a valid BUILD command, the system clears the portion of the screen reserved for the display and the system is ready for the preparation of that display. Proceed as indicated in "Editing a display".

HOW TO MODIFY A DISPLAY

When a display already exists, you may change it in EDIT mode. Simply display the page using the PAGE command word followed by the display name. This puts the selected page on the screen.

Example: PAGE FORM1<cr>

Then enter EDIT<cr> or simply <alt-E>. The system indicates on the top right hand corner of the screen what you are editing. The cursor moves to the top left hand corner of the display (line 4 column 1). You may now proceed with the editing.

THE DISPLAY EDITOR

When editing a display, each displayable character you type is echoed at the cursor location and the cursor moves one position to the right. The <cr> key brings the cursor to the first column of the next line down. You may move the cursor around, within the display area, by using the arrow keys (up, down, left, right).

The display area is from line 5 to line 22 and from column 1 to column 80. If you get to the end of the border, the cursor does not move further. There is no wrap around between lines.

The current cursor location, line and column, is indicated in the upper right hand corner.

The Insert/Replace, controlled by the <Ins> key, as well as the <Del> and <Backspace>, allow

you to do line editing. The <End> key, however, is assigned a new function : it toggles the "Draw Mode" on and off.

When the Draw Mode is on, the bottom left hand of the display shows the pattern selected, along with its identification number for reference. You may change the selection by incrementing or decrementing the pattern identification number, with the <F7> and <F8> keys. Using the arrow keys, you may then trace the selected pattern on the display.

Note that the pattern type 1 is a "space". When you trace with pattern type 1 selected, you simply erase the path of the cursor.

Controlling the draw mode with <End> and the cursor with the arrow keys, you may draw, erase, or simply move around as you see fit.

Pattern type 3 (single line) and type 4 (double line) consist of a family of patterns that are automatically selected, depending on the direction of the movement and the adjoining patterns, to create corners, intersections, tee's, etc. Basically, the pattern is selected to blend in with the lines around it. Single and double lines may be mixed.

These types of patterns may be used to draw rectangles or grids. When drawing a rectangle, you should start somewhere in the middle of any side, rather than a corner. This will illustrate how the patterns are generated and, with a little practice, you will soon master more complicated shapes.

The display editing mode contains two sub modes to allow the editing of the display name and the insertion or modification of dynamic variables within the display.

The draw mode is automatically turned off when a sub mode is requested.

CHANGING THE COLOR

The word COLOR on the top line of the screen show which foreground and background colors are currently selected. You may change those colors with the following keys:

F1	change the foreground color
F2	change the background color
F3	return to the default (white on black)

There are 8 colors to choose from. Whenever you press F1 or F2, the next color is selected.

Note that some combinations may cause the word color (and the text you would be typing) to be invisible (black on black, red on red, etc.)

HOW TO CHANGE THE NAME OF THE DISPLAY

When you edit a display, the name of the display being edited is shown at the top right hand corner of the page preceded by the word "Editing".

If you press <alt-N>, the name of the display is highlighted and the cursor moves to the first character of the name.

You may then change the name to another unique name, by inserting, replacing or deleting characters. After you have composed a new name, press <cr> to enter the change.

If the system rejects the proposed name, you remain in the name editing sub mode. If the name is accepted, then the system automatically returns you to page editing.

You may also abandon the name editing sub mode by pressing the <Home> key.

HOW TO SHOW VARIABLES ON A DISPLAY

To put a variable in a display you must first assign a name to it using the database editing facility.

Move the cursor to the location where you want to show the variable. That position will be the left most or bottom position occupied by the dynamic value or status.

Press <alt-V>. This causes a prompt to appear at the bottom of the page. The prompt (VARIABLE AT LINE y, COL x >), indicates the selected coordinates and waits for a variable specification.

You then type the variable definition using the following format:

name/access type spec color<cr>

name is the name of the variable to be displayed.

/access is an option that allows function key control.

typeis an identifier that selects the method of representation. You may use : BINARY, NUMBER, LZI, REAL, HGRAPH, VGRAPH, LIST, TEXT, VCA, HCA

spec is a further specification that varies with the type you have selected.

color specifies the color attribute to be used for display. This parameter may be a constant or the name of a variable that will evaluate to the color attribute. The color attribute is the decimal value of a binary mask that defines the foreground color, background color, intensity and blink. Refer to the HCA/VCA type below for the details of the relationship between the value of this attribute and the color it produces.

The only parameter that is necessary is "name". All the others are optional, as will be shown below, and are defaulted if omitted.

If / is not present, function key control (arrow up/down, F7/F8) is not allowed on the variable. If / is present, you may append to it the name of an access variable, in which case the function key control will be allowed if and only if that access variable is TRUE. If / is used without the access variable, then function key control will be allowed unconditionally.

If "type" is not specified, the variable is displayed as it appears on the database definition page. That is, a binary input will have its status displayed as OPEN/CLOSED, a binary output will show as OFF/ON, analog and registers will be shown as a real number occupying 11 positions, including the sign and the decimal point, and the number of digits after the decimal specified by the DEC command.

When you specify a variable without a "type", you cannot specify the "color" parameter either. The variable will be displayed with the foreground color currently selected on a black background.

Let us now consider what takes place when a "type" is specified.

NUMBER This specifies that the variable should be displayed as a positive integer. One more argument only is allowed with this specification: a number from 1 to 5 that indicates how many digits should be displayed. If this number is omitted, it is defaulted to 5. Leading zeroes are replaced by spaces.

Unless the color parameter is specified, the currently selected foreground color becomes part of the definition. As it is not possible to select the color (using F1, F2, F3) while defining the variable, you should establish that color before you type <Alt-V>.

Example: PART/ NUMBER 4 CN<cr>

Typing <alt-N> generates the word NUMBER.

LZI This specifies the variable as a Leading Zero Integer. It is used exactly like the NUMBER but when displayed, leading zeroes are not replaced by spaces.

REAL This specifies that the variable should be displayed as a signed decimal number. This specification may be accompanied by a pair of numbers, the first one indicating the total number of positions occupied by the variable (the width), the second one indicating how many of these positions are digits after the decimal point.

The width must be minimum 3 and maximum 11. The number of decimals must be minimum 0, maximum=width-3. In other words, the specification must always allow for a sign, a decimal point and a minimum of 1 numeral.

If the field specification is omitted, the width is automatically 11 and the number of digits after the decimal is as specified at the time by the DEC command. This is particularly useful in situations when the order of magnitude is unknown. If the value became too small, it would be possible to display more decimals without editing the display.

Unless the color parameter is specified, the currently selected foreground color becomes part of the definition. As it is not possible to select the color (using F1, F2, F3) while defining the variable, you should establish that color before you type <Alt-V>.

Example : TEMP1 REAL 7 2<cr>

Typing <alt-R> generates the word REAL.

HGRAPH This specifies that the variable is to be represented by a horizontal bar. Two more numeric arguments must be used with this specification. The first number set the total number of positions to be occupied by the variable. Since this variable representation extends from left to right, the maximum width is limited to the number of positions from the current cursor location to column 80. The second number represents the full scale value of the variable, i.e. the value that will make the variable cover the total width.

The full scale value is limited to 4095. The actual width of the display will be directly proportional to the value of the variable, keeping in mind that the minimum resolution on the screen is one character width. Overflow of the variable is not shown; if the value increases beyond the defined full scale, the display remains at the full scale.

Unless the color parameter is specified, the currently selected foreground color becomes part of the definition. As it is not possible to select the color (using F1, F2, F3) while defining the variable, you should establish that color before you type <Alt-V>. The display of a HGRAPH variable only affects the background color of the area covered, it does not overwrite the foreground.

Example: LEVEL HGRAPH 75 250<cr>

VGRAPH This specifies that the variable is to be represented by a vertical bar graph. The use of this type is similar to HGRAPH, except that the display extends from bottom to top. The maximum height is limited to the number of positions between the current cursor location and the top line (line 5).

Example: LEVEL VGRAPH 10 100<cr>

Typing <alt-V> generates the word VGRAPH.

BINARY This specification must be followed by two words from the following list :

OFF	ON
OPEN	CLOSED
NO	YES
FALSE	TRUE
NON	OUI

The first word will be displayed when the variable is false, the second one when the variable is true. The variable always occupies 6 character positions.

Unless the color parameter is specified, the currently selected foreground color becomes part of the definition. As it is not possible to select the color (using F1, F2, F3) while defining the variable, you should establish that color before you type <Alt-V>.

Typing <alt-B> generates the word BINARY.

LIST The "LIST" format associates the variable with a text array, so that if the integer

value of the variable is n, the variable is displayed as the nth element of the array. The first element of the array is element 0 corresponding to n=0. Each element of the array is a string of characters that you write in the text portion of the data base.

The name of the text array is the name of the variable that corresponds to the first string (element 0).

The "spec" portion for this format consists of the number of elements in the array, the name of the array and the number of characters that the text will occupy on the page.

The number of elements is a constant from 1 to 15. If you indicate 0 as the number of elements, then the display system assumes that your array may contain any number of elements.

The maximum number of characters occupied is 40 since this is the maximum number of characters that the text area of the database can hold.

Example:

Say we have written in the database the following:

```
+-----------------------------------------------------------+
|      NAME  CURRENT  A E  MESSAGE                          |
|                                                           |
| R0155 MODES    0.00     OFF LINE                          |   | R0156        0.00
AUTOMATIC OPERATION                    |
  R0157         0.00     MANUAL OVERRIDE
```

We have also defined the following variables:

```
+-----------------------------------------------------------+
|      NAME  CURRENT  A E  MESSAGE                          |
|                                                           |
| R0111 MODE     0.00                                         |   |
                                                      |


+-----------------------------------------------------------+
|      NAME  CURRENT  A E  MESSAGE                          |
|                                                           |
| R0122 MOTOR_OFF  0.00                                        |   |
                                                        |
```

We are now in the display editor and define the following variable somewhere on the screen:

MODE/MOTOR_OFF LIST 3 MODES 15

Depending on the value of MODE (0, 1 or 2) the variable will be displayed as 'OFF LINE', 'AUTOMATIC OPERA', or 'MANUAL OVERRIDE'.

The Tab or Arrow Up/Down will allow you to select the variable MODE only if 'MOTOR_OFF' is true. If you want the full word OPERATION to appear, change the 15 to 19 in the definition.

You may assign a color to each of the elements of the array by preceding it with "\n" in the text definition page. The character "n" represents the color per the following chart:

1=blue	9=bright blue	G=blinking blue
2=green	A=bright green	H=blinking green
3=cyan	B=bright cyan	I=blinking cyan
4=red	C=bright red	J=blinking red
5=magenta	D=bright magenta	K=blinking magenta
6=yellow	E=bright yellow	L=blinking yellow
7=white	F=bright white	M=blinking white
8=grey		

In the above example,

\JMANUAL OVERRIDE

will use the blink red color when MODE is equal to 2.

This method of assigning a color to the displayed element supersedes the "color" parameter. If you did not precede the text with a "\n", and you did not include the "color" parameter, then the display system will use the white foreground and black background.

If you have indicated as the number of elements a number from 1 to 30, the display system will automatically limit the value of the variable from 0 to n-1 and roll over when decrementing or incrementing the variable with the F7 and F8 keys.

Typing <alt-L> generates the word LIST.

NOTE: The LIST format is not compatible with previous versions of RTES. If you are using a knowledge base developed with an earlier version of RTES, you must :

1. Edit your display to make sure that the list length does not exceed 15.

2. Dump the display using the PD command.

3. Reload your display in the new system using the PL command.

TEXT This specifies that the variable must be replaced by the text associated with the variable. A number must follow to specify the number of positions to be reserved for that text on the display.

Unlike the LIST specification the same text is displayed regardless of the value of the variable.

The color for display is established in the same manner as for the LIST format.

Typing <alt-T> generates the word TEXT.

For example : PRODUCT/ TEXT 25

causes the first 25 characters of the text associated with the variable PRODUCT to be displayed. If this definition is used on user display page, the '/' indicates that the variable may be selected to enter the text.

If a TEXT variable is selected, anything entered on the command line is assumed to be a text to be assigned to the selected variable. If you intend to type a command, make sure a TEXT variable is not currently selected - use <Home> for instance to cancel any current selection, or select a variable that is not a TEXT.

DISPLAY A VARIABLE AS A COLOR ATTRIBUTE

HCA controls the color attribute of a specified number of columns on a line according to the value of the variable.
HCA must be followed by a positive integer indicating the number of columns.

The positive integer value of the variable is interpreted as a color and display attribute according to the following conventions.

1=blue foreground 16=blue background

2=green foreground 32=green background
4=red foreground 64=red background
8=bright 128=blink

To combine colors and/or characteristics, the above numbers must be added up accordingly. For instance, blinking yellow on white background would be obtained with a value of 2+4+16+32+64+128=246

If the sum formed does not contain any of the foreground numbers, then the foreground is black. If it does not contain any of the background number then the background is black. Blue+green+red yields white.

To illustrate the above, a value of 7 is white on black, 112 is reverse video.

VCA controls the color attribute of a specified number of lines on a row according to the value of the variable.

VCA must be followed by a positive integer indicating the number of rows.

The positive integer value of the variable is interpreted as a color in the same way as for the HCA.

ERROR CONDITIONS

If an error is detected in the variable specification, an appropriate message is displayed, and more editing is allowed. If the specification is valid, then the system returns to page editing.

You may also return to page editing, without entering the specification by pressing <Home> or by pressing <cr> with the cursor in the left most position of the displayed specification, immediately after the prompt (>).

MODIFYING A VARIABLE SPECIFICATION

It is often necessary to change the specification of a variable that has been placed on a display. It is also useful to know what the specification is, without necessarily making a change to it.

In such cases, you may place the cursor at the starting coordinates of a variable. This is the left most or bottom position of the highlighted area showing the position occupied by the variable. Once at that position, press <alt-V> and the system will display the prompt as well as the variable specification as entered or as defaulted by the system when the specification made use of default parameters.

Once the line is displayed, it may be edited just like a command line. The revised line must be re-entered by pressing <cr>. Note that any characters at the right of the cursor when <cr> is pressed are ignored. Therefore, if the middle of the line only has been changed, make sure you bring the cursor to the end of the line, by using the <arrow right> or <End> key, before pressing <cr>.

If an error is detected in the revised specification, the original is not erased and more editing is allowed. If the new specification is valid, then it replaces the previous one and the system returns to page editing.

You may also return to page editing by pressing <Home> or by pressing <cr> with the cursor in the left most position of the displayed specification.

SPACE RESTRICTIONS

Variables are not allowed to share the same area of the screen. Once a variable has been specified to occupy a certain area on the page, if another variable is specified that would overlap with a previous one, it is rejected.

Variables are not allowed to extend beyond the page. Should a combination of starting coordinates and specification cause the area to be occupied to extend beyond the boundaries of the display, the specification is rejected.

If an area of the page is occupied by a variable, it is not possible to place characters in that area. A variable, on the other hand, may overwrite any text that was previously placed in its field. It should be noted that the displays generated by the bar graphs do not actually affect the text, but merely highlights the space it occupies.

It is therefore possible to place a text somewhere on the screen and then define a bar graph that will highlight all or part of it. A particularly interesting application of this feature is to specify a binary variable as a bar graph (HGRAPH) and assign to it the width of a certain sentence with a full scale of 1. When the variable is true, the sentence stands out in reverse video.

DELETING A VARIABLE FROM A PAGE

It may be necessary to remove a variable that has been defined at the particular location on the page. In this case, place the cursor at the starting coordinates of the variable to remove, then press <alt-R>. The system will display the specification of the variable and request a confirmation.

Typing "C" (upper case) will confirm the deletion, any other character will just cancel the request to delete and return to page editing.

TERMINATING A DISPLAY EDIT SESSION

While editing the body of the page, you may terminate the edit session by pressing <Home>. If you were editing the display name or entering the specification for a variable and wanted to abandon the edit session, then you would have to press <Home> twice. Once to cancel the <alt-N> or <alt-V>, then once more to return to command mode.

If you are editing a display with a batch file, you terminate the edit session with the HOME command.

EXAMPLES

TOTAL is a numeric variable. We want to display it as a signed number with 3 digits before the decimal and 2 digits after the decimal. Function key control is inhibited.

```
TOTAL REAL 7 2<cr>
```

TYPE is a numeric variable. We want to display it as a 2 digit integer. We want to enter its value using the function key control.

```
TYPE/ NUMBER 2<cr>
```

LOOK is a binary variable. We want to display it as the word NO if false, YES if true. We want to be able to change it from NO to YES and vice versa, using the function key control but only if variable O is true.

```
LOOK/O BINARY NO YES<cr>
```

PC is a numeric variable. We want to display it as a horizontal bar with a full length of 50 characters when its value is 100. PC will be calculated and we do not want to change its value with the function key control.

PC HGRAPH 50 100<cr>

DONE is a binary variable. We want to highlight the sentence "SELECTION MADE" when DONE is true. We place the cursor at the letter "S" and press <alt-V>.

Note that the area to be highlighted is 14 characters long.

DONE HGRAPH 14 1<cr>

COLOR is a numeric variable. It may evaluate to an integer between 0 and 9. A series of 10 consecutive registers starting at COLORS have the following texts associated with them : BLACK, BROWN, RED, ORANGE, YELLOW, GREEN, BLUE, PURPLE, GRAY, WHITE.

The variable COLOR may now be specified as:

COLOR LIST 10 COLORS 6<cr>

The display will show the name of the color corresponding to the value of COLOR (0=black, 1=brown, etc.)

PUMPSTAT is a variable assuming the values 4 or 2. The word PUMP appears on the screen at the coordinates where the following definition is entered:

PUMPSTAT HCA 4<cr>

You may then associate PUMPSTAT with another binary variable PUMP (see RULES) so that PUMPSTAT is 4 when PUMP is false and 2 when PUMP is true. The word PUMP on the screen will then show red if PUMP is false and GREEN if PUMP is true.

LIST OF DISPLAY PAGES

On startup of RTES the first customized display page is displayed and the first operator accessible variable is selected. You may move from one display page to the next using the <Pg Dn> and <Pg Up> keys. The order of the display pages is the order in which they are created

using the BUILD command.

The <Pg Dn> and <Pg Up> key functions can be executed from within a batch file with the use of the PGDN and PGUP commands.

To get a list of the display pages and their order, simply enter the PAGE command without an argument. For example:

PAGE<cr> or <Alt-P><cr>

A specific display page may be called up at any time using the PAGE command.

The format of the command is:

PAGE name<cr>

If you cannot remember the name of a customized display page, you can use the PAGE command to provide you with the list of display pages and then <Tab> to place the name of the display in the argument field following the PAGE command. For example, if you were interested in the third display page, you would press <Tab> 3 times to place the name of the third page as the argument to the PAGE command and then press <cr> to enter the command for execution.

You can then move to the next page using <Pg Dn> or to the previous page using <Pg Up>.

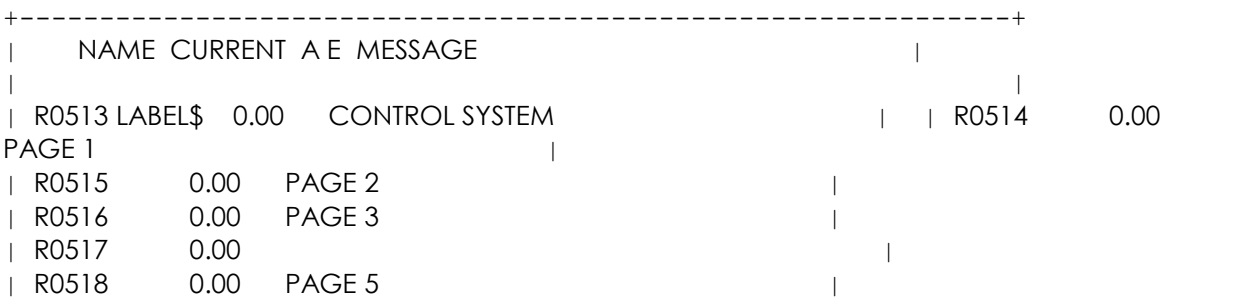
USER DEFINED SCREEN TITLE.

If the variable LABEL$ is found on startup, RTES assumes it defines a text array to be used as title centered on the top line of the screen.

The title is displayed in blue on a cyan background.

The array index is the user page number. RTES places this number in the reserved variable PAGE$ if it is found on startup. (PAGE$=1 for the first page, 2 for the second, etc.). PAGE$ is equal to 0 when there is no user page displayed or it is edit mode. Note the page title is properly selected from the array even if PAGE$ is not defined.

For example:



The top line will display "PAGE 1" when the 1st user page is displayed, "PAGE 5" when the 5th user page is displayed. The 4th user page will not have a title.

If the variable LABEL$ is not defined, RTES displays the product's name at the top of the screen.

USER DEFINED OPERATOR'S INSTRUCTIONS.

Line 22 of the screen is normally reserved for some explanations about the use of function keys. If your configuration file RTES.CNF, does not specify H=0, then line 22 will display how some function keys are normally used.

If you specify H=0 in RTES.CNF, then the content of that line may be defined by two text arrays called BR$ for the right hand 40 columns and BL$ for the left hand 40 columns. If you want to use the variables BR$ and BL$, you must define them in the registers area and re-start RTES so that it recognizes them on startup.

The indexes of the arrays are the values of BR$ and BL$. You may change the values as you wish (command, rule, batch, task). As long as there is a user page displayed, the variables BR$ and BL$ will operate dynamically as if they had been defined as a text array of 40 columns. The color is always yellow.

BR$ and BL$ are treated as dynamic variables, they will be added to the page if less than 96 variables are defined.

USER DEFINED FUNCTION KEYS.

You may associate text arrays to registers called F1$ to F10$ and CF1$ to CF10$. These variables are associated with the function keys F1 to F10 and <Ctrl-F1> to <Ctrl-F10>. The

integer values of each variable is used as the index into its array. When the key is pressed, the array element is passed to the command line. This feature is subject to the following conditions:

The variable must be present on startup of RTES - or before you use that function key for the first time after startup.

The function key interpretation only takes place when a user defined page is displayed.

The value of the index must greater than or equal to 0.

A text must be present at the expected array element.

If one of the above conditions is not met, the function key assumes its default function, if any, or is ignored.

The special character '\' allows you to simulate a <cr> as part of the text array so that you may define a command that executes with a single key stroke. The text ends wherever a '\' is seen. Two consecutive '\\' cause the command to be executed and immediately erased.

Example : We define the following

```
+-------------------------------------------------------------+
|      NAME  CURRENT  A E  MESSAGE                            |
|                                                                 |
| R0097 F3$      0.00     SET X 55\                       |    | R0098        0.00     SET Y
100\                                  |
| R0099         0.00     START PUMP_1\                   |
  R0100         0.00     STOP PUMP_
```

Depending on the value of F3$, pressing <F3> causes the corresponding text to be passed to the command line. In the case of F3$=3, the command line will display STOP PUMP_ and you must type the rest, including a <cr>.

You may also precede the text with '\'. This causes any previous text on the command line to be ignored.

CONTROL OF OPERATOR ACCESSIBLE VARIABLES.

Whenever a new page is displayed by RTES, the first operator accessible variable is selected. This feature may be suppressed by including in RTES.CNF the term F=0.

The next operator accessible variable may be selected using <Tab> or <Down Arrow>. The previous operator accessible variable may be selected using <Up Arrow>. The order of operator accessible variables depends on their position in the display. Rows have precedence over columns.

For example, if you have operator accessible variables at the following coordinates: line 3 column 25, line 4 column 45 and line 3 column 50; the first operator accessible is the variable at line 3 column 25, the second is the variable at line 3 column 50, and the third is the variable at line 4 column 45.

Once a variable is selected you may change its value using <F7> and <F8> provided you have not otherwise defined these function keys, by typing a number on the command line or by using the on-line calculator.

Selection of an operator accessible variable on a page may be cancelled using the <Home> key.

From within a batch file, the HOME command performs the <Home> key function. This method is useful in situations where an operator station does not have the <Home> key available or when using a function key to execute a series of keystrokes and/or commands.

DISK RESIDENT DISPLAYS.

RTES allows up to 16 displays to be memory resident. You may however define more displays and save them on disk.

The page dump command, PD, allows you to store a copy of the current display page into a disk file so that it may be subsequently loaded using the page load command PL or the external page command XP.

The format of the page dump command is:

PD<cr>

This causes a copy of the current display page to be stored in a file named page.PIC where page is the name of the page being stored.

The page load command, PL, allows you to load a display page previously stored to disk by the PD command. Once the display page is loaded, it becomes part of the 16 memory resident

display pages. You must first make the space for that display with the BUILD command - not necessarily with the same name as the page you are loading - and exit from the edit mode. You could also overwrite the current display with the one you are loading. In any case, you must be on a user defined display to use the PL command.

The format of the PL command is:

PL name<cr>

where name is the page name without the .PIC extension

The external page command XP, allows you to display a page previously saved on disk with the PD command, without making it part of your current set of memory resident displays.

The format of the XP command is:

XP name<cr>

where name is the page name without the .PIC extension

The PD and PL commands provide an effective way to export/import pages from one RTES based system to another or to duplicate a page within the same system. Simply copy the .PIC files, make sure that your target system contains the same variable names and load those files onto the new system.

You may also use PD and PL to re-organize your displays. RTES stores the memory image of the display pages in one file named OLEX.PAG. Dump all your displays, using the PD command, then exit RTES and delete OLEX.PAG. When you re-start RTES, it does not contain any displays. You may now create the display pages, using the build command, in the order you want, and load into these pages the original .PIC files using the PL command.

The FPD command operates the same way as PD, but saves the page in a format that is substantially faster to load with the PL or XP command. The PD command stores the definitions of the variables in ASCII format, so that the .PIC file may edited with a word processor. The FPD command saves the definition of the variables in internal binary mode.

RECALL OF LAST DISPLAY.

After you have displayed anything else besides a user defined page, press <Esc> to automatically

return to the last display page you used.

EXTENDED HELP FACILITY

When using a knowledge base, it may be necessary to provide some more information than what is contained in the user's displays. Typically, you may want to explain to the user the choices that are available, any limitations, or maybe why you need the information.

You may associate one text file with each variable. When that variable is selected, pressing <Esc> causes that page to be displayed.

Pressing any key, then returns you to exactly where you were.

CREATING AN EXTENDED HELP SCREEN

You may write your own help files using your favorite word processor, provided you save it as a DOS text file.

You must give this file the same name as the variable it is to be associated with, followed by the extension ".HLP".

The command NOTE allows you to put some text in a file with a given name.

e.g. NOTE MODE.HLP<cr>

The above command will create a disk file called "MODE.HLP". It will erase the screen and allow you to type up to 15 lines of text.

During this creation session, you may use the editing keys : replace/insert, backspace, arrow left, arrow right, delete. The <cr> key moves you to the next line.

After the text is created, you may press <Home> to save it on disk or <Esc> to abandon the session without saving the note file to disk.

The NOTE command may also be used to create other text files for you own use such as a memo pad, comments, etc. An appropriate file name extension is recommended to help you identify those files later on and distinguish them from the ".HLP" files. For instance IDEA.MEM or

BUDGET.CMT would be appropriate file names.

USING THE EXTENDED HELP

Help screens may only be displayed for variables that have been defined in a user page (see Display Generator) with the /(allow) attribute.

When such a variable is selected using <Tab>, <Dn Arrow> or <Up Arrow>, pressing <Esc> causes the note file with the same name as the variable to be read from disk and displayed. You may then return to the original page and selected variable by pressing any key (except <Ctrl-Home> of course, which would return control to the operating system).

If a file with the same name as the variable is not found, the error message "No help available on this subject" is displayed at the error line. Note that help files must be in the current default directory.

When using a batch file to display the extended help file, you use the command HLP. This is the method you would use when the operator station does not have a <Esc> key or when you want to use a function key to display the alert extension file. The HLP command cannot be executed from the command line.

DEFAULT HELP FILE

You may create, using the NOTE command or your favorite word processor, a file called "HELP.HLP" and include in that file some general information about your own application.

When the <Esc> is pressed, while no variable is selected, the system will look for the file "HELP.HLP" and display it. If that file does not exist, the system cause the error message "No variable selected" to appear.

CONTEXT DEPENDENT HELP FILES

When a help file is displayed, it follows the same rules as the report generator. It is therefore

possible to make the content of a help file modify itself depending on the current situation.

Variables may be replaced by the current value or associated text, paragraphs may be conditionally displayed. However, what appears on the screen cannot change dynamically.

MULTI-SCREEN HELP FILES

A help file may consist of several screens, in which case it must be created using your own word processor but should only contain characters that are appropriate to a text file. Formatting characters such as bold, underline, etc. are not supported.

When the system encounters a help file that extends beyond 15 lines, it allows you to display the next screen by typing any key, or return to your work screen by typing <Esc>.

MOUSE SUPPORT

To use the mouse interface you must load your mouse driver before starting RTES. This driver comes with your mouse.

RTES.CNF must contain :

U=n n is the index of the 1st of 9 registers

Register n+1 returns the pixel Y coordinate
Register n+2 returns the pixel X coordinate
Register n+3 returns status of Left button
Register n+4 returns status of Right button
Register n+5 returns status of Middle button (when used)
Register n+6 set to 0 for invisible cursor, set to 1 for visible cursor
Register n+7 returns the character Y coordinate
Register n+8 returns the character X coordinate
Register n+9 returns the identity of the variable located at the cursor, with the following convention:

0	no variable at the location
00001 to 09999	register number
10001 to 19999	digital input number+10000

20001 to 29999	digital output number+20000
30001 to 39999	analog input number+30000
40001 to 49999	analog output number+40000

Regardless of the value of Register n+6, RTES hides the cursor whenever it changes anything on the screen.

CHAPTER TEN - TIME OF DAY SCHEDULER

TIME OF DAY SCHEDULER

It may be desired to turn a variable on or off at specific times of the day. RTES allows you to define such events thanks to a built-in scheduler. Any variable may be controlled several times over a 24 hour period; and several variables may be scheduled to operate according to their own individual schedule. Up to 10 scheduled events can be defined.

DEFINING A SCHEDULE EVENT

The SCHED command allows you to enter or remove an event in the schedule.

The format of the command is

SCHED name hh:mm action<cr> or <Alt-S>name hh:mm action<cr>

where name is the name of the variable to be controlled

hh:mm is the time when the event is to take place, using a 24 hour clock

action is one of 3 choices:

ON	if the variable is to be turned on
OFF	if the variable is to be turned off
REMOVE	if the event is to be cancelled.

When a scheduled event has been defined, it remains in effect until it is removed. If the variable already is at the state dictated by the event, then it remains in that state.

When a variable is the object of one or more scheduled events, the E column on the variable definition page contains an "S" (for schedule). It cannot be evaluated by any other method such as a rule, cascade or loop.

DISPLAY SCHEDULED EVENTS

The command

SCHED name<cr>

causes the events associated with the named variable to be displayed. This display has the following format:

name hh:mm action
name hh:mm action
etc.

If events associated with the displayed variable are added or removed, the display is immediately updated.

PRINT SCHEDULED EVENTS

The command

PS<cr>

causes all the scheduled events to be printed in chronological order.

CHAPTER ELEVEN - TREND RECORDING

OVERVIEW

When the Real-Time Expert System is used for data acquisition and control, a trend recording facility is provided to write to disk at regular intervals the values of selected variables. This feature is controlled by 2 variables named LOGGING and INTERVAL.

Recording takes place when the variable LOGGING is true and the interval in seconds between successive recordings is defined by the variable INTERVAL. These two variables may be located anywhere in the data base and may be controlled manually, by rules, by time of day scheduling, tasks, or combinations of these methods.

The variables to be recorded are defined on line by editing a "trend" table. This table is displayed by the command TREND. Once it is displayed, the EDIT mode allows you to enter, modify, or remove up to 40 variable names in the trend table.

TYPICAL SET UP

A typical set up would go as follows.

Step 1. Type the command TREND<cr>
The trend table content is displayed. It may possibly be empty.

Step 2. Type the command EDIT<cr> (or Alt-E)
The cursor positions itself at the location of the first variable.There are 4 columns of 10 variables each.
The <Tab> key allows you to move from one column to the next.
The <Dn Arrow> and <Up Arrow> keys allow you to move from line to line.

Step 3. Fill the table with the variable names you want to trend, pressing <cr> after each name is typed.

Step 4. Press <Home> to exit the EDIT mode and return to command mode.

Step 5. Define the variables LOGGING ,INTERVAL and EVENT$. Since INTERVAL and INTERVAL must contain a number, it is preferable to use registers for all three.

Step 6. Define an interval in seconds.
For instance SET INTERVAL 30<cr>
or
RULE INTERVAL=60<cr>

Step 7. Define a filename for the events file in the text area of the register EVENT$.

Step 8. Initiate recording by making LOGGING true.
For instance START LOGGING<cr>
The system will now create or open for appending, if it already exists, the events file in the current directory and write to it a header consisting of the date, time, and variable names being logged. Then at the interval of time defined, the values of the variables preceded by the current time are recorded.

Step 9. You may end recording by making LOGGING false.
For instance STOP LOGGING<cr>
When ending a logging session, the systems writes to the events file the date and time when logging ended.

EVENTS FILE

The events file is a text file. Headings are enclosed in quote marks and values are in decimal format with leading zeros suppressed. It is compatible with most popular spread sheets as "imported" data.

The name of the events file is pointed to by the RTES register EVENT$. The register EVENT$ acts as an index to a family of files in the text area associated with the register. The value of EVENT$ is used to select the file into which the events are recorded. If the value of EVENT$ points to an empty text area or the register EVENT$ is not defined, the events will not be recorded to disk.

Example:

The database contains:

```
+------------------------------------------------------------+
```

```
|      NAME  CURRENT  A E  MESSAGE                                    |
|                                                                                |
| -----                                                                  |    | R0012 EVENT$
0.00      EVENT1                                    |
  R0013        0.00      EVENT2
  R0014        0.00      EVENT3
  R0015        0.00      A:EVENT4
```

When EVENT$ is equal to 0, RTES records the events in a file called EVENT1. When EVENT$ is equal to 3, RTES records the events in a file called EVENT4 on drive a. When EVENT$ is equal to 4, RTES does not record the events in any file.

The value of EVENT$ can be determined by the same means as any other variable, i.e. rule, task, operator input, etc.

OTHER SYSTEM INFORMATION ASSOCIATED WITH LOGGING

The following reserved register names will automatically be filled with the appropriate information, when they exist. You may assign these names to any registers and they will be recognized by the Real-Time Expert System and updated every time logging takes place.

DISK_CL Number of available clusters in current drive. If using a floppy, multiply by 1024 to obtain the number of bytes available.
If using a hard disk, multiply by 4096 to obtain the number of bytes available.

LOG_SIZE Number of characters contained in the events file.

ESTART_MTH Month number (1-12) when the last logging session started.

ESTART_DAY Day of the month when the last logging session started.

ESTART_HR Hour of the day when the last logging session started.

ESTART_MN Minute when the last logging session started.

ELAST_MTH Month number of the last logged data.

ELAST_DAY Day of the month of the last logged data.

ELAST_HR Hour of the day of the last logged data.

ELAST_MN Minute of the last logged data.

CHAPTER TWELVE - THE ALERT SYSTEM

OVERVIEW

The alert system provides a method to make the user aware of a specific situation, regardless of what page is on the screen at the time.

To that effect, a variable (used as binary) represents the condition we want to monitor and the text associated with that variable is the message we want to display. We then tell the system if we want the alert condition to be on when the variable is false or true.

When the status of the variable matches the one defined as alert, the message is displayed on the bottom line of the screen, preceded by the variable name and followed by the flashing word "MESSAGE".

Should several alert conditions exist simultaneously, the bottom line will display the most recent alert message.

Alert messages may be acknowledged by the user. When an alert is acknowledged, it allows the next youngest unacknowledged message to occupy the bottom line.

Should all alerts be in the acknowledged state, then the bottom line will continue to display the most recently acknowledged message.

When the condition that causes the alert no longer exists, the alert is said to be "CLEARed" and does not compete any longer for the bottom line.

It is also possible to display the list of all active alerts, acknowledged and unacknowledged. This multi page display is dynamic. If it is on the screen, it will reflect the current activities showing new alerts, acknowledgements and clearing of alerts.

Alert activities may also be directed to the printer and a disk file for logging.

HOW TO DEFINE AN ALERT

Once the variable that represents the alert condition and its associated message exist, an alert may be defined by two methods.

Screen editing method

Display the page containing the alert variable. Enter the EDIT mode and Tab to the "A" column. Use the <Up Arrow>/<Dn Arrow> to place the cursor across from the variable. Then enter :

0	for FALSE = alert condition
1	for TRUE = alert condition
E	for EITHER (change of state)
N	for NO alert condition

The "E" directs any change of state of this variable to the change of state logger.

The "N" simply eliminates a previously defined alert condition for that variable.

ALERT command method

An alert condition may be defined or removed with the ALERT command. The format of the command is :

ALERT name state<cr>

nameis the name of the variable to be associated with the alert condition

state is 0 for FALSE = alert condition
1 for TRUE = alert condition
E for EITHER. This tags the variable for change of state recording.
N for NO alert condition (to cancel a previously existing alert condition)

ALERT ACKNOWLEDGE

The ACK command is used to acknowledge an alert event.

The ACK command may be used in different ways.

ACK<cr>

acknowledges the alert on the line selected. If there no selected line, then it is the youngest alert - displayed at the bottom of the screen that is acknowledged. Pressing <F10> issues this form of the ACK command.

ACK name<cr>

acknowledges the alert you identify by the name of the alert variable. (e.g. ACK TCB1<cr>)

ACK line<cr>

acknowledges the alert currently displayed at the line number you indicate. The argument 'line' is a constant from 4 to 21.

ACK @reg<cr>

acknowledges the alert currently displayed at the line number corresponding to the value of the RTES register 'reg'.

For example: SET ALARM 4<cr>
ACK @ALARM<cr>

acknowledges the alert displayed on line 4.

ALERT STATUS VARIABLES

The Alert System maintains 2 variables: ALARM$ and ACK$. ALARM$ contains the number of variables currently in the alert state. ACK$ contains the number of those variables that have been acknowledged. Therefore, the difference between ALARM$ and ACK$ is the number of unacknowledged alerts currently in the system.

As with other system variables, ALARM$ and ACK$ must be seen on system startup to be used. Therefore, after you define these 2 variables, you must exit RTES and reload it so that these variables are seen upon program loading.

DISPLAY CURRENT ACTIVE ALERTS

The LIST command causes RTES to enter the dynamic alert display mode.

The format of the command is

LIST<cr>

The screen displays all current alert events and maintains the display current. If the screen is full (16 alert lines displayed), then you may press <Pg Dn> to display the next page if any. <Pg Up> returns to the previous page. If there is only one page, then the same page is re-displayed, but CLEARed alerts are eliminated from the display.

RTES may be placed in dynamic alert display mode even if there are no active alerts at the time. It will then display the alert events as they occur.

LOGGING ALERT MESSAGES ON THE PRINTER

If the register PLOGS$ exists when RTES is loaded, its status will control the printing of the alert events.

When PLOGS$ is true, the alert events are printed, provided the printer is not currently printing something else.

In order to use this register, you must define it (see EDITING THE DATA BASE) then exit the RTES and reload it so that the variable is seen upon program loading.

Printing of alert events may be switched on and off by controlling the status of PLOGS$ (operator command, rule, etc.)

Each alert event sent to the printer consists of the current date and time followed by the alert message then followed by the alert status:

MESSAGE, ACK'D or CLEAR

LOGGING ALERTS ON DISK

If the register DLOGS$ exists when RTES is loaded, its status will control the logging of the alert events.

When DLOGS$ is true, the alert events are appended to a disk file pointed to by the register LOG$. If the register LOG$ does not exist, the alert events are not recorded to disk.

The name of the alert file is pointed to by the RTES register LOG$. The register LOG$ acts as an index to a family of files in the text area associated with the register. The value of LOG$ is used to select the file into which the events are recorded. If the value of LOG$ points to an empty text area or the register LOG$ is not defined, the events will not be recorded to disk.

Example:

The database contains:

```
+-------------------------------------------------------------+
|      NAME  CURRENT  A E  MESSAGE                           |
|                                                                 |
|  -----                                                      |    |  R0012 LOG$
0.00      ALARM1                                    |
  R0013         0.00      ALERT2
  R0014         0.00      ALERT3
  R0015         0.00      A:ALARMS
  R0016         0.00
```

When LOG$ is equal to 0, RTES records the events in a file called ALARM1. When LOG$ is equal to 3, RTES records the events in a file called ALARMS on drive a. When LOG$ is equal to 4, RTES does not record the events in any file.

The value of LOG$ can be determined by the same means as any other variable, i.e. rule, task, operator input, etc.

In order to use these registers, you must define them (see EDITING THE DATA BASE) then exit the RTES and reload it so that the variable is seen upon program loading.

Each alert event logged in an alert file consists of the current date and time followed by the alert message then followed by the alert status:

MESSAGE, ACK'D or CLEAR

Logging of alert events may be switched on and off by controlling the status of DLOGS$

(operator command, rule, etc.)

Logging also updates the variable DISK_CL, if it exists at program load time. DISK_CL contains the number of available clusters in the current disk or diskette, thus informing you about the amount of storage room left. Each cluster represents :

1024 bytes of a floppy diskette
4096 bytes of a hard disk

ALERT HISTORIAN

RTES will store the last 500 alert events (alert set, alert clear).

This alert history may be called up on the screen with the command TRACE. The events are shown in reverse chronological order (the last event is at the top of the list).

When the screen shows the alert history, the following keys have a special function:

<PgUp> and <PgDn> rolls the display as permitted by the green arrows at the top and bottom of the display.

<Tab> or <Arrow Down> selects the next alert on the page for recall of the alert extension file using <Alt-X>.

<Alt-X> displays the alert extension file (name.AEX) corresponding to the currently selected alert.

<Esc> returns the screen to user pages.

The alert history may be printed with the command PAH. As with the TRACE command, the events are listed in reverse chronological order.

The RTES command PAH prints the content of the alert historian.

ALERT EXTENSION FACILITY

When an alert condition occurs, it may be necessary to provide some more information than what is contained in the alert message line. Typically, you may want to explain to the user the cause of the alert condition, the actions required, or any precautions that should be taken.

You may associate a text file with each variable. When that variable is in the alert condition and is selected (see 'SELECTING AN ALERT LINE' below), pressing <Alt-X> causes one page (15 lines) to be displayed. Pressing any key in a multiple page alert extension file allows you to display the next page. Pressing <Esc> returns you to exactly where you were. In a single page display, pressing any key returns you to exactly where you were.

In addition to the <Alt-X> function key the AEX command also recalls the alert extension file for the selected variable, but allows more flexibility.

AEX<cr>

recalls the file associated with the alert on the currently selected line, or with the alert displayed at the bottom of the screen if there is no line selected.

AEX name<cr>

displays the alert extension file associated with the alert variable 'name' (name.AEX).

AEX line<cr>

'line' is a constant from 4 to 21. This command format displays the alert extension file associated with the alert on the specified line.

AEX @reg<cr>

'reg' is the name of a RTES register. This command format displays the alert extension file associated with the alert displayed at the line number specified by the value of 'reg'.

Once an alert extension file is displayed, there is a difference between a short file that only occupies one screen and a longer file.

For a short file, pressing any key returns you to the previous page.

For a long file, you must press the <Esc> key to return to the previous display, any other key displays the next page of the file. If you reach the last page, then pressing any key returns you to the previous screen.

If you are using the mouse to control the screen via EXEC statements, then the AEX command gets you to the next page of the file (if any). The HOME command returns you to the previous screen.

CREATING AN ALERT EXTENSION SCREEN

You can write your own alert extension files using your favorite word processor, provided you save them on disk as DOS text files.

It is important to give this file the same name as the variable it is to be associated with followed by the extension ".AEX".

The command NOTE allows you to enter up to 15 lines of text in a file.

e.g. NOTE MODE.AEX<cr>

The above command will create a disk file called "MODE.AEX". It will erase the screen and allow you to type up to 15 lines of text.

During this creation session, you may use the editing keys : replace/insert, backspace, arrow left, arrow right, delete. The <cr> key moves you to the next line.

After the text is created, you may press <Home> to save it on disk or <Esc> to abandon the session without saving the note file to disk.

USING THE ALERT EXTENSION

Alert extension screens may only be displayed for variables that are in the alert state when a user display page or the alert page is being displayed. When the alert message for the variable is displayed on the bottom line of the screen, pressing <Alt-X> causes the alert extension file with the same name as the variable to be read from disk and displayed. You may then return to where you originally were by pressing any key on a single page display or <Esc> on a multiple page display.

If a file with the same name as the variable is not found, the error message "No help available on this subject" is displayed at the error line. Note that alert extension files must be in the current default directory.

Alert variables may also be selected from the display of current active alerts. Use the LIST command to enter the dynamic alert display mode. Use the <Tab> or <Dn Arrow> to select the variable. Press <Alt-X> to display the alert extension file.

When using a batch file to display the alert extension file, you use the command AEX. This is the method you would use when the operator station does not have a <Alt> key or when you want to use a function key to display the alert extension file. The AEX command cannot be executed from the command line.

CONTEXT DEPENDENT ALERT EXTENSION FILES

When an alert extension file is displayed, it follows the same rules as the report generator. It is therefore possible to make the content of an alert extension file modify itself depending on the current situation.

Variables may be replaced by the current value or associated text and paragraphs may be conditionally displayed. However, what appears on the screen cannot change dynamically.

MULTI-SCREEN ALERT EXTENSION FILES

An alert extension file may consist of several screens, in which case it must be created using your own word processor but should only contain characters that are appropriate to a text file. Formatting characters such as bold, underline, etc. are not supported.

When the system encounters an alert extension file that extends beyond 15 lines, it allows you to display the next screen by typing any key, or return to your work screen by typing <Esc>.

CHANGE OF STATE LOGGER

On startup, RTES looks for the variables HCOS$, PCOS$ and DCOS$. If they exist, the change of state logger is activated. A change of state is a transition from false to true or true to false.

Change of state messages are logged in the alert historian if HCOS$ is true; sent to the printer if PCOS$ is true whenever a change of state variable changes from FALSE to TRUE or vice-versa.

When DCOS$ is true, the change of state events are appended to a disk file pointed to by the register COS$. If the register COS$ does not exist, the change of state events are not recorded to disk.

The name of the alert file is pointed to by the RTES register COS$. The register COS$ acts as an index to a family of files in the text area associated with the register. The value of COS$ is used to select the file into which the events are recorded. If the value of COS$ points to an empty text area or the register COS$ is not defined, the events will not be recorded to disk. (see the example for LOG$ above).

Any variable type I, O or R may be designated for change of state logging by placing an "E" in the Alert column.

A change of state message consists of the current time in hh:mm:ss.xx format, the variable name and a 10 character message describing the new status of the variable. This message is defined by the first character in the message field if the new status is OPEN or by the second character in the message field if the new status is closed.

These characters are the digits 0-9 or A-F and they index the first 10 characters in the text fields of the last page of registers. For example, the standard size version of RTES uses registers R2033 to R2048.

Example :

```
+-------------------------------------------------------------+
|     NAME  CURRENT  A E  MESSAGE                             |
|                                                                |
| I0001 LK1      OPEN    E   35 circuit breaker LK1              |   |
                                                |


+-------------------------------------------------------------+
|     NAME  CURRENT  A E  MESSAGE                             |
|                                                                |
|                                                                |
|                                                                |
| R2033                                  OPEN                        |
| R2034                                  NORMAL                    |
| R2035                                  BAD                         |
| R2036                                  CLOSED                          |
| R2037                                  FAULT                     |
```

The word CLOSED will be included in the change of state message when the contact feeding input LK1 is OPEN, and the word TRIP when the contact is closed.

You may use an *' inlieu of the character 0-F that specifies the COS descriptor to tell the system not to log the event.

Example:

```
BREAKER       OPEN    E     23MAIN CIRCUIT BREAKER
```

The above will log the event with descriptor 3 when BREAKER changes from open to close and descriptor 2 when BREAKER changes from closed to open.

```
BREAKER       OPEN    E     *3MAIN CIRCUIT BREAKER
```

The above will log the event with descriptor 3 when BREAKER changes from open to close and will not log the event when BREAKER changes from closed to open.

If an alert variable is changed to a change of state variable, an entry is made in the COS log indicating the current status of the variable.

You may choose to include the date in the change of state message. To do so, include in RTES.CNF :

D=1

You may override the location of the change of state descriptors by including in RTES.CNF the parameter Q=id of the first (of 16) registers that will define the change of state descriptions.

The TRACE command allows the display of the historical file in reverse chronological order. The format of the command is

TRACE<cr>

When the historian is on the screen, arrows pointing up and/or down inform you that there are more records above or below. The <PGUP> and <PGDN> keys are active to scroll the records one at a time. The PGUP and PGDN commands perform the same function.

SELECTING AN ALERT LINE

When alerts are displayed (using the LIST, or TRACE commands), you may use the <UP ARROW> and <DOWN ARROW> keys to move a pointer ">" along the left edge of the screen and select one the lines.

This marker selects the alert line as default for the ACK, and AEX commands.

The SEL command does the same thing but provides more flexibility.

Using the SEL command,

SEL<cr>

you simply select the next line. If the last line was selected, then the next line is the first line again.

You may also use the command

SEL line<cr>

'line' being a numeric that specifies the line number. This command places the selection mark (>) on the line you specified, provided there is an alert line at that location. The first line is line 4, the last one line 21.

For example : SEL 4<cr>.
selects the 1st alert line

You may also use the command

SEL @reg<cr>

'reg' being the name of a RTES register whose value is the line number we want to select.

For example :

SET ALARM 4<cr>
SEL @ALARM<cr>
selects the 4th alert line on the screen

Pressing the <HOME> key cancels any current selection. The HOME command also cancels any

current selection.

CHAPTER THIRTEEN - BATCH COMMAND FACILITY

OVERVIEW

A disk file may contain a series of operator commands to be executed sequentially on demand. This is particularly handy when you want to set the values of certain parameters, limits, alert messages, specific to some particular condition while retaining the rest of the knowledge base.

Such a file, created ahead of time using a word processor or the NOTE command (if it does not contain more than 15 lines) or created automatically using the FILE and XFILE commands, is called a batch file.

Each of these files has a unique file name and is executed by the command

GET file_name<cr> or GET @variable_name

The batch file name could be a path such as B:SETUP1.CMD thus extending the reach of the system beyond the current default directory.

When you use the GET command to retrieve the data back from disk, the system opens the file named as the argument to the GET command and executes each line in the file as a command.

Batch command files can be used to update another site running the same application. You could create a file containing the commands required and send it to the site on a floppy disk. The user at the other site could then use the GET command to enter the new commands instead of entering them at the keyboard.

BATCH FILE FORMAT

A batch file contains one command per line. Each line follows the syntax appropriate to the command used. For instance, a batch file may contain

```
NAME R17 A
NAME R18 B
NAME R19 C
RULE A=B+C
TEXT A RESULT
```

SET B 12
SET C 20.5

Any operator command that could be issued from the command line is allowed in a batch file. You are urged to use your own discretion when including editing commands such as EDIT or RULE.

BATCH FILE EXECUTION

When a GET command is issued, the content of the specified file name is assumed to be a series of operator commands. These commands are executed concurrently with other executing tasks.

If a line is erroneous because of a syntax error or an invalid argument or any other reason, the system simply ignores it and goes on to the next line.

The lines are executed in the sequence they are read. In the above example, if the RULE line preceded one of the NAME lines and the variables mentioned did not exist, the RULE command would be ignored.

The commands issued in a batch file must be compatible with existing conditions. For instance, if we added to the above batch file the command

SET A 20

it would be without effect, since A is defined as B+C and A would remain 32.5.

Only one batch file may be executed at any given time. A GET command issued while a batch file is executed, causes the error message "Task in progress". An interesting consequence is that if a batch file contains a GET command, that command is always ignored.

You may define a register named READING$. When this register exists, RTES sets it to 1 when it starts executing a batch command file and to 0 when the execution is over. This register may used as a flag to make sure that a batch file has been executed before going to the next one.

The GET command has two formats:

GET filename or GET @variable

The 'GET filename' format lets you access the batch file directly. The 'GET @variable' format

lets you access the batch file indirectly from a text array associated with the variable. The value of the variable is an index pointing to the batch filename which is the first word in the text string for each text message in the array.

AUTOMATICALLY CREATED BATCH COMMAND FILES

A particular use of the batch files is to enter a set a data specific to a case, into the knowledge base. Such batch files consist of a series of SET commands and occasionally some TEXT commands pertaining to a set a variables.

For example, let us consider the variables PRODUCT, MASS, VOLUME, TEMP.

You may be using or studying several products that each have some specific mass, volume and temperature, but using the same knowledge base.

It is possible to have the following batch file called "RUBBER"

```
TEXT PRODUCT RUBBER # 1234
SET MASS 1.2
SET VOLUME 3.6
SET TEMP 72
```

and another batch file called "WOOD"

```
TEXT PRODUCT ACACIA WOOD
SET MASS .8
SET VOLUME 5
SET TEMP 122
```

Such files may be automatically created using the FILE command and a single pre-defined format file named "DATA.FMT" or using the XFILE command and various format files named "name.FMT".

For the above example, DATA.FMT would contain

```
*PRODUCT
MASS
VOLUME
TEMP
```

You may then set the data and use the command

FILE file_name<cr> or XFILE DATA file_name<cr>

to save that data along with the appropriate TEXT or SET command formats in the named batch file. Note the presence of the "*" preceding PRODUCT in DATA.FMT which causes the TEXT command to be used in lieu of the SET command.

Once the various files are created using the FILE command, you may recall them into the knowledge base by the using the GET command. For instance :

GET RUBBER<cr>
or
GET WOOD<cr>

name.FMT FILE FORMAT

When the FILE command is executed the system creates and opens a file on disk with the name of the argument to the FILE command. It then reads each line of the file DATA.FMT to determine what data is to be stored.

When the XFILE command is executed the system creates and opens a file on disk with the name of the second argument to the XFILE command. It then reads each line of the file name.FMT to determine what data is to be stored.

The file DATA.FMT contains the description of which commands are to be stored in the file specified in the FILE command. The file name.FMT contains the description of which commands are to be stored in the file specified by the XFILE command.

To store a variable value simply write the name of the variable. For example, TEMP. This is translated into the following batch command line

SET name value

where: name is the variable name

value is the current value of the variable

To store the text associated with the variable, simply write the name of the variable preceded by an asterisk. For example, *PRODUCT. This is translated into the following batch command line

TEXT name text

where: name is the variable name
text is the text currently associated with the variable

The batch filename may also be specified indirectly using a variable and a text array. Using this method the formats of the FILE and XFILE commands are:

FILE @variable or XFILE name @variable

where: variable points to the beginning of the text array and contains the value of the index into the array. The first word in each text message in the array is the name of the batch file. The maximum value of the variable for indexing is 255.

The FILE and GET or XFILE and GET mechanisms may be a used to transfer data and/or results from one application to another. One application stores the data using the FILE or XFILE command. The other application then retrieves the data using the GET command. Of course, the names of the variables must be the same in each application.

The FILE and GET or XFILE and GET mechanisms may also be used to store and obtain the values used in various 'recipes' in the same application.

DEFAULT FILE ARGUMENT

The FILE and XFILE commands may be used without specifying a batch file name.
This is acceptable only if a GET command was previously used. The FILE command then uses the file name specified in the last GET command.

For instance:

GET RUBBER<cr>

you may change the data at this time

FILE<cr> or XFILE DATA<cr>

the new data is saved in "RUBBER".

CHAPTER FOURTEEN - REPORT GENERATOR

OVERVIEW

The Real-Time Expert System will produce printed reports using a text created by a conventional word processor, customized to reflect conditions set up by the knowledge base and current values of the variables imbedded in the report.

IMBEDDED VARIABLES

Within a document, a variable is introduced by the sequence @@, followed by a specification and ending with @.

The specification is written in accordance with the syntax explained in the Display Generator section of this manual. However, the HGRAPH and VGRAPH formats are not available for report generation.

Any invalid variable specification will cause the variable to be skipped at print time.

Example:

The data base contains the variables NAME and AGE.

Currently, NAME=0 and AGE=21.

The text "Frank Sinatra" is associated with the variable name.

The document "SINGER" created with a word processor contains:

> The singer @@NAME TEXT NAME 20@ is @@AGE NUMBER 2@ years old.

Typing the command

PRINT SINGER<cr>

will cause the following to be printed:

The singer Frank Sinatra is 21 years old.

The command:

PRINT @REPLIST

Will use the file name in the text area of the register REPLIST augmented (indexed) by the integer value of REPLIST.

PRINT ON CONDITION

You may cause certain portions of the document to be printed or skipped depending on a given expression being true or false.

The syntax is as follows :

[[condition]text.....]

A double open square bracket introduces the statement. The condition follows and is terminated by one closing square bracket. The text contained until the next closing square bracket will be printed if the condition is true or will be skipped if the condition is false.

Example :

The document "REPORT" contains :

[[RAIN&COLD]The weather is poor for the season.]

If the data base contains RAIN=TRUE and COLD=TRUE, the command

PRINT REPORT<cr>

will cause to be printed

The weather is poor for the season.

If the condition contains a syntax error, then the text is skipped up to the closing bracket, as if the condition were false.

THE PRINT COMMAND

Once a report has been created, you may print it with the command

PRINT report_name<cr>

The report_name specified is opened and 'spooled' to the printer as it is interpreted by the report generator. Any required form feeds must be included in the report as an ASCII 12 decimal.

The report_name may consist of a path such as B:SUMMARY.DOC, so that it does not necessarily have to be in the default directory.

While a report is being composed, any subsequent PRINT commands cannot be executed. It causes the error message 'Task in progress' to be displayed.

You may specify the pages of a report to be printed. The format of the PRINT command to do so is:

PRINT report_name first last<cr>

Example:

PRINT PRODUCT.REP 3 5<cr>

causes pages 3 to 5 of the report PRODUCT.REP to be printed.

PRINTING THE KNOWLEDGE BASE

The command summary details the P* commands that control the printing of the variable definitions, rules, scheduled events and task listings.

The knowledge base print command requests are automatically queued by the system and executed in the order in which they have been received. Up to 8 (eight) commands may be queued. If the print queue is full and a knowledge base print command is received, it causes the

error message 'Print queue full' to be displayed.

When the system prints the knowledge base, it automatically performs a form feed every 56 lines and prints an appropriate header at the top of the page. The system also performs a form feed at the end of each print job.

You should position the paper and set the top of form prior to printing.

PRINTING IN PROGRESS FLAG

If the register PRINT$ exists, the system will set it to 1 while composing the report and set it back to 0 when it has finished and is ready to accept another PRINT command. This variable may be used to coordinate the printing of several reports consecutively from within a task. For example, a task may
contain the following

```
.
.
25 X PRINT SUMMARY.DOC
26 IF [PRINT$] 26
27 X PRINT RESULTS.DOC
.
.
```

In the above, the task will wait until the system has completed the first print request before sending the next PRINT command.

The PRINT command will still operate without the register PRINT$ being defined, but will give no indication to the user as to when it has completed the request. It should be noted that since output to the printer is buffered, the system may be ready to accept the next PRINT request although the printer is still busy.

Warning:

The system only sets PRINT$ to 1 when it starts formatting a report and back to 0 when it is done. It is up to the user to make sure that some other action does not set PRINT$ to a misleading value.

EXTERNAL PRINT FONT DEFINITION

RTES by default assumes that the attached printer can only print the basic set of ASCII characters. The External Print Font Definition feature allows those users with printer that have more sophisticated features to make better use of their printers.

The characters sent to the printer may be translated using a table prepared by the user. The table must be in a file named name.PDF.

The file contains a series of 4 numbers in decimal, separated by commas, spaces and/or line feeds. The first number is the ASCII value to be translated, the other 3 numbers are the ASCII characters sent to the printer in place of the first one.

If a character is only translated into one or two characters, the remaining numbers in the set must be 0's.

Any text between a semicolon (;) and the end of line, is considered a comment.

Up to 255 translations may be accommodated. (Total 1020 numbers)

The file is loaded by the command FONT. The format of the command is :

PDEF name<cr>

Once a .PDF file is loaded, it entirely replaces the existing default translations.

The default character translation table is:

```
09,09,0,0          ;HORIZONTAL TAB (must be defined)
12,12,0,0          ;FORM FEED (must be defined)
13,13,10,0         ;LF AFTER CR
242,103,116,101    ;greater than or equal
243,108,116,101    ;less than or equal
251,115,113,114    ;square root
130,101,8,39       ;e accent grave
131,97,8,94        ;a circonflex
136,101,8,94       ;e circonflex
133,97,8,96        ;a accent `
138,101,8,96       ;e accent `
```

```
140,105,8,94        ;i circonflex
141,105,8,96        ;i accent `
147,111,8,94        ;o circonflex
149,111,8,96        ;o accent `
150,117,8,94        ;u circonflex
151,117,8,96        ;u accent `
135,99,8,44         ;c cedille
128,67,8,44         ;C CEDILLE
```

Any of the above, if needed, must be included in the user defined .PDF file.

NOTE: Any character with an ASCII value less than 32 or greater than 122 must be defined, otherwise it is printed as *. We suggest that the first 3 lines of the above table are included in any user defined PDF file. The next 3 lines are used to print 'gte', 'lte' and 'sqr'. You may replace them by the appropriate ASCII code, if your printer has the proper symbol.

ABORT CURRENT PRINT JOB

The STP command allows you to abort the current print job. Any spooled print commands are then executed.

The format of the command is:

STP<cr>

CHAPTER FIFTEEN - COMMUNICATIONS INTERFACE

INTRODUCTION

RTES is specifically designed to communicate with external devices such as data acquisition and control subsystems, PLCs, other computers, or other instruments.

RTES can communicate with up to 5 external devices at one time via 5 ports labelled : PORT0, PORT1, PORT2, PORT3, PORT4.

PORT1 to PORT4 are usually associated with the physical 'COM' ports - or serial communication adapters - of your computer. PORT0 is designed to address other types of devices that do not utilize those serial communication ports.

Various programs called 'drivers' are available. Each driver has been designed to match the communication characteristics and protocol of the device it is associated with.

For instance, the driver called "OPTOMUX" allows RTES to communicate with an OPTOMUX network.

Since the RTES device drivers are external memory resident programs, they must be loaded (or installed) before RTES is started up. At that time, you decide which of the available ports you are going to assign to that particular driver. They also remain in memory until the computer is rebooted. Therefore, in order to remove a driver, you must reboot your computer system.

The purpose of the device driver is to exchange data between the RTES variables (I, O, A, Y, R) and the device. The I/O assignment - which specific RTES variables will correspond to which elements (channels, registers) of the device - is determined by a configuration file that you define. The configuration file also determines in which direction the data will move (RTES to device or device to RTES), as well as other aspects of the interface such as the baud rate. When RTES is started up, the configuration files for the installed drivers are read and the and the various operating parameters are established. You may modify the configuration files and thus change these operating parameters, in most cases without shutting down your system using the CONF command.

The following is a partial list of the currently available device drivers:

AB2	Allen Bradley PLC-2 (serial)
AB3	Allen Bradley PLC-3 (serial)

ADAC	A variety of bus mounted I/O cards (parallel)
AB5	Allen Bradley PLC-5 (serial)
IOPLEX	DuTec's IOPLEXER (serial)
MB	Mitsubishi (serial)
MODBUS	Modicon (serial ASCII)
MOD_RTU	Modicon (serial RTU protocol)
MODPLUS	Modicon Modbus Plus - requires SA85 driver
OMRON	Omron PLC (serial)
OPTOMUX	OPTOMUX (serial)
PAMUX	PAMUX (parallel)
SIEMENS	Siemens (serial)
SIXNET	Digitronic's SIXNET (parallel)
TI300	TI300 series PLC (serial)
TI500	TI500 series PLC (serial)

Several more specialized device drivers are available to interface with scales, high speed counters, motion controllers, batch controllers, DCS's, etc. RTES also supports multi-port serial port expanders and telephone diallers. Contact RTS Inc. for any special requirements.

LOADING A RTES DRIVER

Each driver is a separate program that you load into memory using a DOS command. For instance, you type

OPTOMUX<cr>

and the OPTOMUX.EXE file - or program - is loaded into memory. The drivers are designed so that they remain in memory until you re-boot your computer.

When you load a serial device driver, it is attached to the RTES PORT1 unless you specify otherwise. In other words, PORT1 is the default port.

If you want to assign the driver to PORT2 or PORT3, you must specify which port you want to use on the DOS command line.

For instance:

OPTOMUX 2<cr>

will load the OPTOMUX driver and assign it to PORT2.

OPTOMUX 1<cr> is equivalent to OPTOMUX<cr>

When you load a driver, it identifies itself by displaying a message on the screen.

For instance : 'OPTOMUX Driver Ver. 5.2 on PORT2'

If such a message does not appear, it means that the port you want to use is not available. You may already have loaded the driver or you may have loaded another driver on that port.

Drivers that do not use serial communication are usually assigned to PORT0 and cannot be assigned to the other ports. Drivers that use serial communication cannot be assigned to PORT0.

CONFIGURATION FILES

RTES recognizes which drivers are in memory and initializes them according to the information found in a text file - that you create - called the 'configuration file'. There must be a configuration file for each of the ports used. The contents of each of the configuration files will depend on the driver that is going to use it. Typically, you will define the PC port address you will be using (if other than standard), the baud rate (if applicable) and at which locations the data will be exchanged between RTES and your device.

The configuration files have the same name as the assigned ports with the CNF extension, i.e., PORT0.CNF, PORT1.CNF, PORT2.CNF, and PORT3.CNF.

For instance, if we have loaded the driver OPTOMUX and assigned it to PORT1, then you must create a file named PORT1.CNF where you will write the information concerning the OPTOMUX network that you are using, and how you want the physical points (channels) to be mapped to the RTES database.

You can use the word processor of your choice - or the RTES 'NOTE' command - to create and edit your configuration files. Each configuration file must be saved as a DOS TEXT FILE and be given the appropriate name (PORTx.CNF).

Detailed descriptions of the contents of each configuration file are found in the appropriate sections of this chapter, or in the .DOC files provided with each driver. The following addresses common aspects of all driver configuration files.

A configuration file contains a series of 'definitions' separated by spaces, commas or line feeds. Each definition uses the format

x=y

where x is a letter representing a given piece of information.

y is the information, consisting of 1 or more numbers.

For instance, 'B' represents the baud rate. If you want your communication to operate at 1200 bauds, then you must include in your configuration file :

B=1200

To include comments in your configuration file, first type a semi-colon (;), then the comment, then a line feed (<cr>). When a configuration file is read by RTES, the comments are displayed at the bottom of the your screen.

For instance, you may include in your configuration file:

B=1200 ;the serial link is set at 1200 bauds.

Or you may prefer:

;Baud rate:
B=1200

REGISTER INDEX VERSUS REGISTER NUMBER

Within the configuration files, we use an index - or id - to identify the variables. The first register (R0001) is identified as 0. Register R0002 is 1, etc. This convention has been adopted to make the configuration files consistent with the numbering method used in the vast majority of devices you would be interfacing with.

PARAMETERS

In addition to the I/O assignment, several parameters may be defined in the configuration files. These are summarized blow :

Serial type drivers:

baud rate
time-out constant
modem support
serial port address
interrupt request line
block enable option
port failure

Other drivers:

I/O port address
real time clock refresh
time-out constant

If a parameter is missing in the configuration file, a default value is assumed.

BAUD RATE

'B' is the letter representing the baud rate parameter.

Examples:

B=300
B=9600

Baud rates may range from 150 to 38400. If your configuration file does not define a baud rate, the default 19200 baud is assumed.

TIME-OUT CONSTANT

The time-out constant defines how long you want RTES to wait for each character in the response from the attached device. This time is in milliseconds is M=0 (default) in RTES.CNF.

Time-out constants may range from 1 to 64000. If your configuration file does not define a time-out constant, a default of 512 is assumed.

Examples:

```
T=300
T=32000
```

MODEM SUPPORT

RTES serial drivers control the request-to-send RS232 signal. It is turned high when sending data and turned low when not sending data. This allows "multidrop" applications, where several transmitters are wired in parallel and only one of them is enabled at any given time.

'M' is the letter representing the modem support parameter type.

The default value for M is 0 indicating that the clear-to-send signal does not need to be checked. A value of 1 indicates that the clear-to-send signal should be set before sending occurs.

The clear-to-send signal is not normally checked before the driver transmits. If your application requires that you can only send if the clear-to-send is present, then you must include in your configuration file :

```
M=1
```

If M is greater than 1, then it defines the index of a register used for the operation of a 'dial-up' modem. This feature is supported by some selected devices drivers. See DIAL-UP MODEMS and the discussions on the individual device drivers for details.

SERIAL PORT ADDRESS AND INTERRUPT REQUEST LINE

When you load your driver, the PORT number you specify on the command line selects the default serial port address and interrupt request line (IRQ) that the driver will use.

The default port selections and interrupt request lines are:

```
PORT1 defaults as port 1016 (3F8 hex) and IRQ 4 (COM1)
PORT2 defaults as port 760  (2F8 hex) and IRQ 3 (COM2)
PORT3 defaults as port 824  (338 hex) and IRQ 2 (COM3)
```

PORT4 defaults as port 568 (238 hex) and IRQ 5 (COM4)

You may however change these defaults by including in your configuration file which port address and IRQ line you want to use.

'P' is the letter representing the port.

'Q' is the letter representing the IRQ.

The port value you specify must be the decimal representation of the port number.

Example:

P=1024 (for 400hex)

The interrupt request line numbers may range from 2 to 15.

Example:

Q=7

As with all parameters, the port and IRQ selected in the configuration file take effect after the driver has been loaded in memory and does not affect the name of the configuration file. The file name remains PORTn.CNF with the 'n' you entered on the command line when you loaded the driver. (PORT1.CNF if you did not specify).

BLOCK ENABLE OPTION

Most device drivers allow you to decide wether or not a given block of information will be exchanged with the device at a given time. This is done by associating with that block a register. When that register is true, then communication takes place.

In most cases, you may want to control that register on and off. But for some applications, you may want that register to switch automatically to false, once the communication has taken place so that the transfer of data only takes place once..

You control this option with the 'C' parameter in the configuration file.

C=0 means that you control on and off the communication enable registers.

C=1 means that you want the driver to turn the communication enable register off after the exchange has taken place.

The default is C=0.

The block enable register may be any RTES 'R' register except the first one (index 0). Using 0 as block enable, cancels the block enable option for that block and communication goes on as if the block enable register were always true.

PORT FAILURE

E=n specifies n as the index of a register that will be set to true when an error is detected at the serial port (framing, parity, interrupt switched off). If you do not specify E in the configuration file, such errors are not reported.

I/O PORT ADDRESS

This only applies to drivers loaded on PORT0.

'B' is the letter representing the port address.

If your configuration file does not define a port address, a default value will be assumed.
The default port addresses will vary with each type of device. For example, the default port address for PAMUX is 256 (100H). The port address is specified as a decimal number.

Example:

B=1024 (for 400hex)

REAL TIME CLOCK REFRESH

This feature is only available for drivers loaded in PORT0. You may use RTC.EXE to perform this function if you are not using PORT0 for interfacing.

'R' is the letter representing the real time clock refresh.

The default value is 0 indicating that no refresh from the real time clock is to occur or that there is

no real time clock in the system.

The value 1 allows the driver to get the current time from the real time.

Example:

R=1

I/O ASSIGNMENT

In addition to the above general characteristics parameters, the configuration file must include a description of how RTES associates its variables with the data it sends to and receives from the device (data acquisition and control subsystem, PLC, etc.).

RTES variables are assigned to specific channels on the external devices by specifying clusters. Each cluster is a series of contiguous RTES variables (generally up to 16) attached to a series of contiguous I/O channels of the same type on the device.

For example, we may want to receive the temperature readings from the thermocouples connected to channels 0 through 3 on the OPTOMUX board address 55 (decimal) on the analog inputs A001 through A004.

There are four standard cluster types corresponding to the four types of physical I/O points. They are I for binary input clusters, O for binary output clusters, A for analog input clusters, and Y for analog output clusters.

Other cluster types may exist depending on the purpose of the device driver and/or the type of device. For example, the OPTOMUX driver uses G clusters to define counters, a special feature of the OPTOMUX's B1 brain boards.

I/O assignments have the general form:

T=p C1, C2, ... Cp

where T is the cluster type (i.e., I, O, A, or Y)
p is the number of cluster definitions to follow
C1, C2, etc. represent the cluster definitions which will be specific for each different driver

Cluster definitions specify where in the RTES database the physical I/O are mapped. For example, this is how you would specify that the first digital input point on a device is mapped to the first

point on the 2nd RTES digital input page.

Depending on the device type that is attached to a port, the number of values specified in a cluster definition will vary. A cluster definition usually includes the following parameters:

first RTES variable in block
quantity of variables in block
position of first variable on device
RTES alarm register
RTES communication enable register

First RTES Variable in Block

This number represents the index to the point location in the RTES data base for the first variable in the block. The index is 1 less than the actual RTES point number. For example, if you were defining an I block and the index was 11, the first variable's value would be placed in I0012.

Quantity of Variables in Block

The value specified here represents the number of consecutive points/channels in the cluster (usually a maximum of 16).

Position of First Variable on Device

The value specified here is device specific. It conforms to the device's method of addressing points. Sometimes more than one number is required. For example, when communicating with an OPTOMUX network, you must specify the base or card address as well as the position on the card.

RTES alarm register

The value specified here is the index of the RTES register that will be set if communication with the device fails. The use of this parameter requires that the protocol for the device must have a mechanism that allows the device driver to detect a communication failure.

RTES communication enable register.

Any RTES register in except the first one (which has an index 0) may be used as communication enable register. Using 0 is a convention to indicate that you do not want to define a register for this function and that the communication defined by this particular block is to take place

continuously.

If you define a non-zero register index, then you must turn that register on (1 or odd value) for the communication to take place. Depending on the 'C' parameter in this configuration file, that register will remain on until you switch it off (C=0), or will be switched off automatically after the block of information has been transferred (C=1).

For example, the OPTOMUX driver cluster definition requires 6 parameters in the following order:

FIRST, QUANTITY, BASE, POSITION, ALARM, ENABLE

Together BASE and POSITION indicate the location of the points in the OPTOMUX network. BASE is the address of the OPTOMUX base or card (0-255) and POSITION is the position of the first channel on the OPTOMUX base.

Example:

I=1 3,6,255,5,5,0

The above line means that there is 1 digital input cluster; the first point starts at I4 (RTES digital input number 4); there are 6 points in the cluster; the input points are found on the OPTOMUX base with address jumpered to 255; and the first point is located at channel 5 on the OPTOMUX board. In other words, the statuses of channels 5 to 10 inclusive on OPTOMUX base 255 will be reflected in the RTES digital input points I4 to I9. Any communication problems will be flagged in RTES register R6. RTES will read this input cluster continuously, since the enable register index is 0.

PROTOCOL ANALYZER

RTES has a built in protocol analyzer. It displays, on demand, the data exchanged via the serial ports. Each serial port can be accessed individually. This protocol analyzer is toggled on and off with the Ln commands, where n represents the driver PORT number. L1 is used with the PORT1 driver, L2 is used with the PORT2 driver, L3 with the PORT3 driver and L4 with the PORT4 driver. You enter the command to turn the protocol analyzer on, you enter the command again to turn it off.

When the protocol analyzer is toggled on, RTES displays the outgoing data in normal video (white on black). The incoming data is displayed in reverse video (black on white).

Note that the Ln command only control the display. The communication itself is not affected by these commands. Also, the protocol analyzers are automatically turned off whenever you use the CONF command.

This feature is very handy for debugging communication problems.

The Ln command also tells the device driver to display its identification on line 2 of the screen.

The L0 command also makes the device driver installed on PORT0 identify itself. There is usually no protocol analyzer capability associated with drivers on PORT0.

DIAL-UP OPTION FOR RTES DEVICE DRIVERS

The dial-up option allows to communicate with a device via a dial up modem using the "AT" commands.

This option is activated by including in the configuration file PORTn.CNF, the definition

M=n where n is from 2 to 2044

The n parameter defines the first of a block of 4 RTES registers used as follows:

HOOK you turn this register on to start a dial up sequence you turn it off to hang up.
If the connection fails, this register is turned off by the driver.
You may place in the message area of this register, a modem setup string of your choice. If you leave this message area blank the modem uses the default string: ATEQL3MV1DP

TEL you set this register to the register number that contains in the message area the telephone number to be dialled.

TALK this register is turned on by the driver when the connection is established. Note that both HOOK and TALK must be on to communicate through the telephone connection.

PASS this register contains in the message area the password to be used for a security device. This password is used if the value of PASS is 1 (or odd).

Example :

The configuration file contains : M=16 (the first register is R0017)

```
R0017    HOOK     0    (optional modem setup string)
R0018    TEL      32       (points to NUMBER)
R0019    TALK     0
R0020    PASS     0    SECRET
:
:
R0032    NUMBER   0    5551212
```

COMMONLY USED DEVICE DRIVERS

Depending on what device you interface with, the format of the clusters and some features will be different. The following sections deal with the most common device drivers. You may find more information about a given driver in the documentation accompanying the software.

ADAC 4000 SERIES

Use PORT0.CNF for configuration.

A/D DATA ACQUISITION BOARDS

Each 'A' cluster defines a set on consecutive analog inputs

The definition of the cluster consists of 7 numbers.

REG, QTY, BOARD, CHANNEL, GAIN, ALA, POL

REG is the index of the first RTES A register where the value will be written.

QTY is the number of channels to read

BOARD is the board base address in decimal

CHANNEL is the starting channel to be read (0-31)

GAIN is a number that defines the programmable gain : 1, 2, 4 or 8 (see ADAC manual).

ALA is the index of an RTES R register that will be set to 1 if the board does not respond after a time delay defined by T and to 0 if it responds.

POL is 0 for unipolar (straight binary), 1 for bipolar (two's complement)

DIGITAL OUTPUTS

Each series of digital outputs is defined by a 'O' cluster.

The cluster definition consists of 4 numbers

REG, QTY, BOARD, POS

REG is the index of the first of the RTES O registers

QTY is the number of consecutive channels.

BOARD is the port address where the digital outputs are located - in decimal.

POS is the position of the first output : 0 to 7

D/A CONVERTER BOARDS

Each set of consecutive channels is defined by a 'Y' cluster.

The cluster definition consists of 4 numbers.

REG, POS, BOARD, CHANNEL

REG is the index of the 1st RTES Y register to be used

QTY is the number of consecutive channels.

BOARD is the address of the board in decimal.

CHANNEL is the starting channel to read on the board (0 to 31) of consecutive channels to read.

PAMUX

This driver uses PORT0.CNF as configuration file.

The base address of the AC28 device is defined with the B= parameter. The default is 256 (or 100Hex).

Each cluster is specified by 4 numbers for digital clusters and 5 numbers for analog clusters. They are:

FIRST, QUANTITY, BASE, POSITION, ALARM

FIRST: first RTES point/channel in the cluster (0 - 256)

QUANTITY: number of consecutive points/channels in the cluster (1-8 for digital, 1-16 for analog)

BASE: the byte offset where the cluster is defined (board address)

POSITION: Position of the first point/channel within the base (0-7 for digital, 0-15 for analog)

Note: When using quad modules, the position can only be 0 OR 4.

ALARM: RTES alarm register index. Used for analogs only.
When using analog boards, RTES must gain access to the on board dual ported memory. Should access not be granted due to board failure or power off, the register defined as alarm is set to 1.
This register must not be present in the digital clusters.

I clusters read binary inputs from the PAMUX boards to I registers
O clusters write binary outputs from O registers to the PAMUX boards
A clusters read analog inputs from the PAMUX boards to A registers
Y clusters write analog outputs from Y registers to the PAMUX boards.

PCLAB

INPUT/OUTPUTS

Each cluster is specified by 4 numbers :

FIRST, QUANTITY, BASE, POSITION

FIRST: first RTES point/channel in the cluster (0 - 256)

QUANTITY: number of consecutive points/channels in the cluster (1-8 for digital, 1-16 for analog)

BASE: the decimal address of the PCLAB board

POSITION: Position of the first point/channel within the board

I clusters read binary inputs from the PCLAB boards to I registers
O clusters write binary outputs from O registers to the PCLAB boards
A clusters read analog inputs from the PCLAB boards to A registers
Y clusters write analog outputs from Y registers to the PCLAB boards.

COUNTERS

Counters are defined as G blocks. You may define up to 64 counters.

Each counter is defined by a set of 4 numbers:

register, address, mode, counter#

register is the index (number-1) of the 1st of 3 RTES registers that will handle the interface with the counter.

The 1st register is used to read the counter.

The 2nd register is an off/on switch that you turn on to read the count and off to stop reading the count.

The 3rd register is a reset switch. You set it to 1 when you want to write a '0' to the counter. The driver sets this register back to 0 when it has
zeroed the counter.

address is the decimal base I/O port address of the pcl-720 board.

mode is the number 0,1,2,3,4 or 5 that selects the operating mode of the counter.

counter# is the number 0,1,or 2 indicating which of the 3 counters on the board you are using.

Example:

We want register R0017 to get the count from counter 0 of board address 512 (200h). R0018 will the control register. R0019 will be the reset register. The counter will operate in mode 2.

Our configuration (PORT0.CNF) file will contain:

G=1 16,512,2,0

RTES will contain the following:

R0017 COUNT 4567.00
R0018 ST_SP 1.00
R0019 RESET 0.00

SIXNET

This driver interfaces RTES with the 6IBMN Sixnet network controller card. It assumes that the 6IBMN has been configured using the Sixnet utility IOTEST_L filename.CFG -RUN

This driver uses PORT0.CNF for configuration. The following parameters are allowed.

DIGITAL INPUTS

Each 'I' cluster may consist of up to 8 contiguous digital input channels.

Each cluster is defined by a set of 5 numbers:

first, quantity, station, position, alarm

first is the index of the first RTES I register of the group.

quantity is the number of I registers (and input channels) in the cluster.

station is the sixnet station number where the input channels are located. (This is used only to establish whether the station is on line).

position is the bit position of the first input channel in the bit map - that you defined using the SIXNET set-up program.

alarm: RTES communication alarm register. (0 - 2047)

NOTE: Inputs channels are arranged in groups of 8, arranged as bytes. When you define the position and the quantity for a given cluster, all the bits used must remain within the same byte.

DIGITAL OUTPUTS

Each 'O' cluster may consist of up to 8 contiguous digital output channels.

Each cluster is defined by a set of 5 numbers:

first, quantity, station, position, alarm

first is the index of the first RTES O register of the group.

quantity is the number of O registers (and output channels) in the cluster.

station is the sixnet station number where the output channels are located. (This is used only to establish whether the station is on line).

position is the bit position of the first output channel in the bit map - that you defined using the SIXNET set-up program.

NOTE: Outputs channels are arranged in groups of 8, arranged as bytes. When you define the position and the quantity for a given cluster, all the bits used must remain within the same byte.

ANALOG INPUTS

Each 'A' cluster may consist of up to 16 contiguous analog input channels.

Each cluster is defined by a set of 6 numbers:

first, quantity, station, position, alarm, enable

first is the index of the first RTES R register of the group.

quantity is the number of R registers (and input channels) in the cluster.

station is the sixnet station number where the input channels are located. (This is used only to establish whether the station is on line).

position is the position of the first input channel in the input channel list - that you defined using the SIXNET set-up program.

alarm: RTES communication alarm register. (0 - 2047)

enable: RTES communication enable register. (1 - 2047)

ANALOG OUTPUTS

Each 'Y' cluster may consist of up to 16 contiguous analog output channels.

Each cluster is defined by a set of 6 numbers:

first, quantity, station, position, alarm, enable

first is the index of the first RTES R register of the group.

quantity is the number of R registers (and output channels) in the cluster.

station is the sixnet station number where the input channels are located. (This is used only to establish whether the station is on line).

position is the position of the first output channel in the output channel list - that you defined using the SIXNET set-up program.

alarm: RTES communication alarm register. (0 - 2047)

enable: RTES communication enable register. (1 - 2047)

INTERFACE SPEED

T=x

xis a constant that determines the number of machine cycles between i/o scans. It must be established experimentally depending on the computer used and the number of clusters defined.

FORMATS FOR I,O AND Y CLUSTERS

FIRST, QTY, BOARD, POS, ALA, ENABLE

CLUSTER FORMAT FOR A

FIRST, QTY, BOARD, POS, ALA, ENABLE, PROBE

FIRST the index of the first RTES variable in the cluster

I clusters read binary channels into I registers
O clusters write O registers to binary channels
A clusters read analog channels into A or R registers (depending on the PROBE parameter)
Y clusters write Y registers (raw values) into analog output channels.

QTY the number of variables in the cluster

BOARD the address of the OPTOMUX board (decimal)

POS the position of the first channel to be transferred

ALA the index of the RTES alert register

ENABLE the index of the RTES enable register

PROBE analog inputs only:
If 0, request the counts from the OPTOMUX board and use them as raw values for the 'A' variables.
If >0, request the EXTENDED TEMPERATURE READINGS for the probe type corresponding to the value of PROBE and place them in the 'R' registers.

COUNTERS

Each counter is defined by a G cluster. You may define up to 64 counters.

The format of a counter cluster is :

register, 0, address, position, alarm, enable

(c)1992-2005 FAI Designs Inc.

register: the index of a RTES register where the count will appear. The next 2 registers are used to control the counter:

set the first register to true to count.
set the second register to true to clear the count. (it returns to false when count is cleared)

0: a dummy argument to maintain the cluster format.

address: hardware address of the OPTOMUX board where the counter is.

position: position (0-15) of the counting input on the board.

alarm: index of the register that will indicate a non or bad response from the counter.

enable: index of the register that controls the communication with this counter. If the register is on (1 or odd value), RTES will attempt communication with this counter, the counter will be ignored otherwise. Any register, starting at R0002 may be used. If you use 0, a enable register does not apply: RTES communicates with the counter continuously. This register is not affected by the C option, the enable register is never returned to 0 automatically.

Example:

We want to count the false to true transitions of the input located at position 5 of board address 200. The count will be read into register R0150. R0151 will be used to start/stop the counter, R0152 will be used to clear the counter. Communication alarm will be annunciated at register R0005. There is no enable register.

The configuration file will include:

G=1 149,0,200,5,4,0

MITSUBISHI SERIES A PLC
USING COMPUTER INTERFACE AJ71C24.

The PLC serial communication port must be set at

8 DATA BITS, NO PARITY, 1 STOP BIT
CHECKSUM OPTION SELECTED
WRITE DURING RUN ALLOWED

Set switches as follows

SW11 OFF (RS232)
SW12 ON
SW13 PER BAUD RATE DESIRED
SW14 ------
SW15 ------
SW16 OFF
SW17 OFF
SW18 OFF
SW21 ON
SW22 ON
SW23 OFF
SW24 OFF

PROTOCOLE : MODE SETTING SWITCH ON POSITION "1"

I/O ASSIGNMENTS

I CLUSTERS DEFINE TRANSFER FROM M AREA TO RTES I VARIABLES
O CLUSTERS DEFINE TRANSFER FROM RTES O VARIABLES TO M AREA
A CLUSTERS DEFINE TRANSFER FROM D AREA TO RTES R VARIABLES
Y CLUSTERS DEFINE TRANSFER FROM RTES R VARIABLES TO D AREA

Each cluster is 6 bytes : 1ST, QTY, STA, ADR, ALARM, ENABLE

1ST: The first RTES variable in the cluster (0-255)
QTY: The number of variables/points transferred (1-16)
STA: Station address (PLC) (0-31)
ADR: First point in the plc area (0-255)

(c)1992-2005 FAI Designs Inc.

ALARM: Index of the RTES register that will be set if a communication error occurs with this cluster. (512 - 2047). (The first 512 RTES registers are not allowed for this function)

ENABLE: RTES communication enable register. (1 - 2047)

A-B PLC2 DRIVER

This driver operates in BCC protocole with :

no parity
1 stop bit
8 data bits

RTES variables are assigned corresponding registers on the A-B PLC network by specifying clusters. Each cluster is a series of contiguous RTES variables (UP TO 16) attached to a series of contiguous A-B PLC registers at the same PLC address.

There are 4 types of clusters :

Iclusters receive bit information from the A-B PLC registers and place them in the I variables.

Oclusters send bit information from the O variables to the A-B PLC registers

Aclusters receive integers from the A-B PLC registers words and use them as value for the R variables.

Yclusters send the values of the R variables as integers to the A-B PLC registers.

I and O cluster definitions consist of 6 numbers:

FIRST, QUANTITY, STATION, WORD, ALARM, ENABLE

A and Y cluster definitions consist of 7 numbers

FIRST, QUANTITY, STATION, WORD, ALARM, ENABLE, TYPE

FIRST: Index of the first RTES point/channel in the cluster (0 - 255)

QUANTITY: Number of consecutive points/channels in the cluster (1 to 16)

STATION: The station number selected at the PLC - in decimal.

WORD: Address of the first data word - in decimal.

FOR DIGITAL I/O'S, EACH 16 BIT PLC WORD REPRESENTS UP TO 16 RTES POINTS. THE LEAST SIGNIFICANT BIT CORRESPONDS TO THE LOWEST NUMBERED

RTES POINT IN THE CLUSTER.

FOR ANALOG I/O'S, EACH PLC WORD REPRESENTS AN RTES INPUT OR OUTPUT ONLY THE 12 LEAST SIGNIFICANT BITS ARE USED.

ALARM: Index of the RTES register that will be set if a communication error occurs with this cluster. (0 - 2047).

ENABLE : Index of any RTES enable register.

TYPE: 0 for transferring decimal numbers. 1 for transferring as AB counters (BCD).

A-B PLC5

This driver operates in BCC protocole with :

no parity
1 stop bit
8 data bits

There are 6 types of clusters.

Iclusters receive data from a PLC word and place each bit of the word in a RTES I register. The rightmost bit goes to the first register in the cluster.

Oclusters send the value of a series of up to 16 O registers as one word to the specified PLC word.

Aclusters receive the values of PLC words as 12 bit integers and uses them as raw values for the specified RTES A registers.

Yclusters send the raw values of the Y registers to PLC words

Gclusters receive the values of PLC words as 16 bit integers and uses them as values for the specified RTES R registers.

Sclusters send the values of the R registers to PLC words as 16 bit integers.

Cluster definitions consist of 7 numbers:

FIRST, QUANTITY, STATION, FILE NUMBER, OFFSET, ALARM, ENABLE

FIRST: Index of the first RTES point/channel in the cluster (0 - 255)

QUANTITY: Number of consecutive points/channels in the cluster (1 to 16)

STATION: The station number selected at the PLC - in decimal.

FILE NUMBER: The 'B' file number to used for transfer.

OFFSET: Offset within the file of the first data word - in decimal.

ALARM: Index of the RTES register that will be set if a communication error occurs with this cluster. (0 - 2047).

ENABLE : Index of the RTES enable register.

This driver supports dial-up modems.

MODICON MODBUS PLUS

Note: you must load the SA85 driver as well as this driver.

Each cluster is specified by 6 numbers:

FIRST, QUANTITY, SITE, POSITION, ALARM, ENABLE

FIRST: First RTES point/channel in the cluster.

QUANTITY: Number of consecutive points/channels in the cluster (1 TO 16)

SITE: Address on the modbus unit (0 - 255)

POSITION: Position of the first point/channel on the MODBUS unit (0-64000)

ALARM: Index of the RTES register that will be set if a communication error occurs with this cluster. (512 - 2047). (The first 512 RTES registers are not allowed for this function)

ENABLE: RTES communication enable register. (1 - 2047)

In the following, C1,C2, Etc represent the cluster specifications an p is the number of clusters.

To define the binary input clusters from registers 40xxx, to RTES 'I' registers

I=p C1,C2,C3,....,Cp

To define the binary output clusters from RTES 'O' registers to 40xxx registers,
O=p C1,C2,C3,....,Cp

To define the analog input clusters from 40xxx registers to RTES 'R' registers
G=p C1,C2,C3,....,Cp

To define the analog output clusters from RTES 'R' registers to 40xxx registers
S=p C1,C2,C3,....,Cp

To define the analog input clusters from 30xxx registers to RTES 'R' registers
A=p C1,C2,C3,....,Cp

MODICON MODBUS RTU

The serial port of the PLC must set at:

even parity
1 stop bit
8 data bits

HOST OPERATION

Each cluster is specified by 6 numbers:

FIRST, QUANTITY, SITE, POSITION, ALARM, ENABLE

FIRST: First RTES point/channel in the cluster.

QUANTITY: Number of consecutive points/channels in the cluster (1 TO 16)

SITE: Address on the modbus unit (0 - 255)

POSITION: Position of the first point/channel on the MODBUS unit (0-64000)

ALARM: Index of the RTES register that will be set if a communication error occurs with this cluster. (0 - 2047).

ENABLE: RTES communication enable register. (1 - 2047)

BINARY INPUTS

I clusters transfer bit information from the PLC to I registers.

The 'POSITION' argument selects where the data comes from:

00xxx reads data from 'coils' (00xxx)
10xxx reads data from status inputs (10xxx)
30xxx reads data from 30xxx holding registers. Each bit corresponding to one I RTES register.
40xxx reads data from 40xxx holding registers. Each bit corresponding to one I RTES register.

BINARY OUTPUTS

O clusters transfer bit information from O registers to PLC registers.

The 'POSITION' argument selects where the data is written at the PLC:

00xxx writes data to 'coils' (00xxx)
40xxx writes data to 40xxx registers. Each O RTES register corresponding to one bit of the 16 bit word.

ANALOG INPUTS

G clusters transfer 16 bit word information from the PLC 40xxx registers to R registers.
A clusters transfer 12 bit word information from the PLC 30xxx registers to the raw values of A registers.

ANALOG OUTPUTS

S clusters transfer 16 bit word data from RTES R registers to PLC 40xxx registers

Y clusters transfer 12 bit word data from the raw values of RTES Y registers to PLC 30xxx registers.

SLAVE OPERATION

The MOD_RTU driver operates as a slave when it is not active as a host. To inhibit activity as a host, you must not have any clusters defined in the PORTn.CNF file, or you must turn off the 'enable' registers of all the clusters that are defined.

Under these conditions, the driver may receive unsolicited messages from a host using a subset of the Modbus RTU protocol. The Modbus commands supported are:

The configuration file PORTn.CNF may include:

N=id to specify the slave ID. (1 to 255). Default is 255.

This driver does not support broadcast commands.

The following commands are supported:

01 REQUEST COIL STATUS

RTES responds with the status of the O registers specified in the command. Coil 1 is O001, coil 2 is O002, etc.

02 REQUEST INPUT STATUS

RTES responds with the status of the I registers specified in the command. Input 10001 is I001, input 10002 is I002, etc.

03 READ OUTPUT REGISTERS

RTES responds with the numeric values (signed integers) of the R registers specified in the command. Register 40001 is R0001, register 40002 is R0002, etc.

04 READ INPUT REGISTERS

RTES responds with the numeric raw values (0 to 4095) of the A registers specified in the command. Register 30001 is A001, register 30002 is A002, etc.

05 FORCE SINGLE COIL

RTES sets the specified O register at the value specified by the least significant bit of the first data byte received.

06 PRESET SINGLE REGISTER

RTES sets the specified R register at the value specified by the data word received.

07 READ EXCEPTION STATUS

RTES responds with the value of the first 8 O registers (O001 to O008)

15 FORCE MULTIPLE COILS

RTES sets the specified O registers at the specified values.

16 FORCE MULTIPLE REGISTERS

RTES sets the specified R registers at the specified values.

17 REPORT SLAVE ID

RTES returns two bytes. Both are FFhex.

This driver supports dial-up modems.

OMRON

The serial port at the PLC must be set up for:

7 data bits
even parity
2 stop bits

There are six types of clusters:

Read IR area : "I" clusters - to RTES I registers
Read HR area : "G" clusters - to RTES R registers
Read DM area : "A" clusters - to RTES R registers
Write IR area : "O" clusters - from RTES O registers
Write HR area : "S" clusters - from RTES R registers
Write DM area : "Y" clusters - from RTES R registers

Each cluster is defined as follows:

First, Qty, Station, Register, Alarm, Enable

First : 1st RTES Registers

Qty : Number of register read/written (maximum 16)

Station : OMRON station number

Register: 1st register in the OMRON PLC

Alarm : RTES ALARM Register (must be larger than 512)

Enable:RTES communication enable register. (1 - 2047)

Note 1: IR always transfer 16 1-bit registers

Note 2: Write operations do not work in RUN mode.

SIEMENS PLC

This driver is compatible with the "CP525" communication interface.

The serial port at the PLC must be set up for:

8 data bits
even parity
1 stop bit

Each cluster is specified by 6 numbers:

FIRST, QTY, BLOCK, OFFSET, ALARM, ENABLE

FIRST: Index of the first RTES variable of the cluster

QTY: The number of variables in the cluster (maximum 16)

BLOCK: Block number in the PLC where the information will be read or written

WORD: word offset within the block

ALARM: Index of the RTES register that will be set if a communication error occurs with this cluster. (512 - 2047). (The first 512 RTES registers are not allowed for this function)

ENABLE: RTES communication enable register. (1 - 2047)

There are 4 categories of clusters :

"I" clusters receive bit information from the plc and place them in the I variables.

"O" clusters send bit information from the O variables to the plc

"A" clusters receive 12 bit integers from the plc words and use them as raw value for the A variables.

"Y" clusters send the raw values of the Y variables as 12 bit integers to the plc words.

In the following, Cx represents a cluster definition and p the number of clusters

BINARY INPUT CLUSTERS:
I=p C1,C2,C3,....,Cp

BINARY OUTPUT CLUSTERS:
O=p C1,C2,C3,....,Cp

ANALOG INPUT CLUSTERS:
A=p C1,C2,C3,....,Cp

ANALOG OUTPUT CLUSTERS:
Y=p C1,C2,C3,....,Cp

TEXAS INSTRUMENTS 500 SERIES

The serial port of the PLC must be set for :

Parity : ODD
Stop bits: 1
Data bits: 7

This driver uses the following clusters:

IRead bits from a PLC V word into the I registers.
OWrite bits from the O registers into a PLC V word.
ARead analog values from the PLC V words into the A registers, as raw values.
YWrite analog values converted from the Y registers into the PLC V words.
GRead numeric values from the PLC V words into the R registers.
SWrite numeric values from the R registers into the PLC V words.

Note 1: Analog values in the PLC are 11 bits starting at bit n. (The n least significant bits are not used). The number of bits n depends on the analog input modules used. This driver allows you to skip those bits to suit your application when reading and writing analog values.

Note 2: Numeric values transferred are positive integers or real floating point numbers depending on the TYPE selected. Floating point numbers use two consecutive V words.

Note 3: Depending on the speed of the PC you are using, you may have to specify M=1 in RTES.CNF

Note 4: Binary data must start at a PLC word boundary. Binary data is always transferred in 16 bit chunks. When reading, RTES only uses the quantity of bits specified in the cluster definition. When writing, 16 bits are sent - 0 filled when not used. Bits are numbered from right to left: the least significant bit in the PLC word corresponds to the lowest number I or O register in the cluster.

The I, and O clusters are defined by 5 numbers:

FIRST, QTY, POS, ALA, ENABLE

The A and Y clusters are defined by 6 numbers:

FIRST, QTY, POS, ALA, ENABLE, PAD

The G and S clusters are defined by 6 numbers:

FIRST, QTY, POS, ALA, ENABLE, TYPE

FIRST is the 1st RTES register in the cluster (index 0 to 2047)

QTY is the number of variables to transfer.
For I and O blocks, QTY is 1 to 16 (bits)
For A and G blocks, QTY is 1 to 15 (words) - 7 if real
For Y and S blocks, QTY is 1 to 7 (words) - 3 if real

POS is the offset of the first V variable (0 to 1023)

ALA is the RTES timeout alert register (0 to 2047)

ENABLE is the RTES enable register (1 to 2047) - or 0 for continuous scan.

TYPE is 0 for integer or 1 for real numbers (floating point)

PAD is the number of rightmost bits that TI reserves and are not part of the analog value, including the overrange flag. (See your TI manual for details).

This driver supports dial-up modems.

TROUBLESHOOTING COMMUNICATION PROBLEMS

In case of difficulty with your communication, you may find here some suggestions on various causes.

If RTES is not sending data, it could be due to one of the following reasons:

1. the CTS signal is expected (M=1) and not present
2. there was no configuration file loaded
3. the configuration file did not have any clusters defined
4. all enable registers are off
5. an error was detected in the configuration file

Use the NOTE command to view the contents of the PORTn.CNF file. If the file is empty or does not contain any cluster definitions, put the appropriate cluster definitions in it and force RTES to read the contents of the configuration file with the CONF command.

If the configuration file had clusters defined, check to see if the modem control (M=1) has been defined. If you have M=1 in your configuration file then the driver waits for CTS to be asserted before sending data. This signal must be provided by your device or the serial port card. Check your hardware to see how it is configured. If the CTS signal is not provided, remove the M=1 from the configuration file and force RTES to reload the configuration file with the CONF command.

If you see only data in normal video, you will know that there is data being transmitted to but no data being received from the device. This could be due to one of the following reasons:

1. wrong cable
2. device not properly set up
3. defective (or duplicate) serial port
4. configuration does not match device

You should check to make sure that your device is connected to the proper serial port on your computer (check both ends of your cable), that your device is powered up, and that your device is operating in the mode required to communicate with a computer host. You can check to see if your device is configured with the correct set of physical characteristics - baud rate, parity, stop bits, data bits. Reconfigure your device if necessary or change the configuration file.

You can check out the integrity of your serial port by jumpering the transmit and receive pins together at the computer end. You should see each byte that is being transmitted being echoed. If

you don't, replace your serial port card.

To check the integrity of the communication cable disconnect the cable at the device end and jumper together the transmit and receive lines of the cable. If your cable is intact, you will see each byte that is being transmitted echoed back. If you don't see any data being received, replace the cable. If you see data being received but it's not the same data being transmitted, check that the cable is grounded properly.

You may also use the software supplied by the manufacturer to test your device. For example, you can use OPTOSCAN to checkout your OPTOMUX equipment.

When checking your configuration file, make sure that your device specifications are correct. For example, do you have the proper device or card addresses for the physical I/O channels. If you are trying to address a device that does not exist in your network, you will not get a response.

If you see data displayed in reverse video only, we must assume that your device is acting as the master and expecting RTES to respond at the appropriate time. If RTES is not responding check that there is a configuration file for the port in your default directory and that it contains the proper cluster definitions. In the case where your RTES is a slave and the Host/Slave network, your configuration file should contain some clusters with block id's that are defined in the host's configuration file.

If you see data displayed in both normal and reverse video and the link is not operating (point status/values are wrong), you would have to examine the data and determine whether it "makes sense" according to the communication protocol specified for that device. For example, if the data coming looks like "garbage", the device's physical characteristics may not be the same as what the driver expects. Check the baud rate, parity, number of stop bits, and number of data bits being used by the device. Make any necessary changes in the configuration of the device and/or configuration file. If the physical characteristics are alright, you may need to reduce the baud rate or increase the time-out constant (the T parameter).

In some cases, you may be getting a NAK (negative acknowledgement) indicating that you are requesting information from or trying to write data to an invalid location on the device. Check your configuration file.

The baud rate may be too high for the length of the cable or the type of environment (It may be too "noisy").

PERFORMANCE TIPS

Many factors can affect the performance of your communication: type of computer, number and type of I/O points on the system, the type of device (parallel vs serial), number of devices being controlled or monitored, baud rate at which the devices communicate, communication protocol used, distribution of devices, etc.

One of the main indications of a performance problem is frequent communication alarms. Quite often these communication alarms are due to the baud rate being too high for the type of line you use. Reducing the baud rate will reduce the number of communication errors encountered.

Sometimes communication alarms will occur when the computer is too fast for the device and doesn't allow enough turn around time to receive the response from the device. Increasing the time-out constant will give the device more time to respond before RTES decides that the device is not responding.

This often happens when you network several RTES systems running on different speeds and/or types of computer. For example, an AT host communicating with a XT slave will require a large larger time-out constant.

If several devices are multi-dropped on a single serial port, you may notice a slow update time for individual points. The scan time can often be reduced by distributing the devices to be scanned over two or three of the serial ports.

REAL TIME CLOCK REFRESH - Serial Drivers

If you are not using a PORT0 driver, you can install the program RTC.EXE to obtain a Real Time Clock Refresh.

This driver reads periodically (every minute) the on board clock and refreshes the DOS clock.

It should only be used if your computer has a real time clock on board and you are not using a parallel (non-serial) interface driver.

To load this driver, execute "RTC.EXE" before loading RTES, with the DOS command :

RTC<cr>

NETWORK DRIVERS - EIA 422/485

Two or more RTES systems can be networked using the HOST network drivers.

The HOST driver is used to make one RTES node initiate exchange of register values with other RTES nodes that do not initiate any communication (they are in SLAVE mode).

Communication may use a serial port EIA 232C, if there is one slave only, or a multidropped EIA 422/485 network.

A variety of commercially available 232 to 422 converters are suitable for multidropped dropped network applications.

Transmission of data is done by blocks. Each block within the network is identified by a unique ID number. A block defines a series of registers. The blocks are defined in the PORTn.CNF file. Each block in the host's configuration should have a corresponding block in only one of the slaves configuration.

SLAVES do not initiate any communication. They wait for a message, and execute it only if they recognizes the block id in the message as one that is defined in their own PORTn.CNF.

A block used to receive information is in the "I" or "G" category. A block used to send information is contained in the "O" or "S" category.

"I" and "O" blocks exchange binary information. An even number is transmitted as a 0 and an odd number as a 1. When used as binary data, only the integer part of a number is considered and negative numbers are considered 0. "G" and "S" blocks exchange full floating point numbers.

Each output block (O or S) in the host must have a corresponding input block (I or G) with the same length and the same id in one and only one of the slaves.

The general form of a block is :

register, quantity, id, alarm, enable, type

register: is the first register index of the block where the data is to be transferred.

quantity: is the number of registers to transfer. Once the first register is established, the quantity may make the block overlap on the next page.

The maximum number of register values that may be transferred is 16 per block.

id: is the identification number of the block. Valid id's are 0 to 255 for each block type.

alarm: is the index of the register to be set in case a communication error or time-out occurs when this particular block is transferred.

enable: is the index of the register that will control transmission.

type: the type of RTES register type to transfer data to or from according to the following chart

0	R
1	I
2	O
3	A engineering unit
4	Y engineering unit
5	A raw value
6	Y raw value

The 'alarm' and 'enable' parameters are only used by the HOST but are present - for consistency - in all configuration files. The assumption is that once the configuration files are built, and provided the interconnection wiring allows it, it would suffice to activate the 'enable' registers at one of the nodes to make it the HOST. The 'enable' parameter can only be 0 at one of the nodes, which would make that node the HOST.

You may define up to 255 blocks of each type.

Example

Say we want the host to receive 8 register values from a slave and place them at R0009 to R0016. The host communicates to the slave(s) via its COM2 at 9600 baud. Communication errors will set R0001. There is no enable register. In PORT2.CNF, we will have:

```
B=9600
I=1 8,8,6,0,0,0
```

Note that the block id has been chosen arbitrarily as 6. In order for a slave to respond, it must have a block of the form :

```
O=1 32,8,6,0,20,0
```

In the above example, whenever the host requests the values of the registers in block 6, it will

receive from this slave, the values of the 8 registers at R0033 (R0033 to R0040). The host will then place those values at R0009 to R0016.

A host sends register values to a slave using a similar set up with the O block at the host and the corresponding I block at the slave.

This driver supports dial-up modems.

USING THE NETWORK DRIVER TO COMMUNICATE WITH ANOTHER INTELLIGENT SYSTEM.

You may substitute the host driver with your own software program to make RTES communicate with other computer based systems. The following description of the protocol used by the RTES host/slave network drivers is intended for computer programmers who wish to write such programs.

The physical characteristics of the interface are

serial
1 start bit
8 data bits
1 stop bit
no parity

The baud rate is set by the B parameter in PORTn.CNF

The following characters have a special significance:

<STX> (decimal 02)
<DLE> (decimal 16)
<ETX> (decimal 03)

Initiating a transfer - send data

The node acting as HOST processes the O and Y blocks according to the control parameter and the status of the command register. For each block, it sends the following bytes:

<STX>
S or O (the upper case character)
i (the block id)
d.....d (data)
<DLE>
<ETX>
<bcc> (an error checking byte)

After the following message is sent, it expects the following response:

<STX>
A
i (the block id)
<DLE>
<ETX>
<bcc>

Initiating a transfer - request data

The node acting as HOST requests the input blocks automatically according to the control parameter in the configuration file and the status of the command register. For every block, it sends the following message:

<STX>
G or I
i (the block id)
<DLE>
<ETX>
<bcc>

After sending the above message, it expects the following response:

<STX>
A
i
d....d (Data)
<DLE>
<ETX>
<bcc>

Data representation (d....d)

Floating point numbers (G and S blocks)

The d....d field consists of a series of 4 bytes per register value.

The first byte transferred is the least significant. The most significant byte is transferred last:

:4th byte:3rd byte:2nd byte:1st byte:

The four bytes represents the value of the register in 32 bit floating point format:

The most significant bit (leftmost) is the sign

0 means positive
1 means negative

The next 7 bits represent a hex exponent with "excess 64".

The rightmost 24 bits are 6 hex digits representing the left justified mantissa. The "decimal" point is assumed between the exponent and the mantissa.

A null is always represented by 4 null bytes.

Binary values, (I and O blocks):

Two bytes are sent, representing 16 registers, right justified, 0 filled to the left if less than 16 registers were requested or sent.

The two bytes are sent twice for further error checking.

DLE as data.

Whenever a DLE occurs in the data field, it is repeated.

Error checking byte. (<bcc>)

The last byte of the message is a "checksum" character. It is obtained by performing an exclusive-or of all the bytes in the message, between the starting STX and the DLE.

The STX and the DLE are not included in the error checking byte.

Duplicate DLE's are not included in the error checking. A DLE in the data field is sent twice but it is xor'ed only once. The error checking byte is null at the beginning of each error checking byte calculation.

RTES - ARCNET DRIVER

The ARCNET installable driver allows computer nodes running RTES on an 'Arcnet' network to communicate.

Any RTES may request from any other RTES on the network the current values or status of groups of registers.

ARCNET may be installed on PORT1, PORT2, PORT3 and/or PORT4. This assigns it to one of the available driver tasks and determines which configuration it will use. It does not associate it in any way with a COM port.

For example, when you load ARCNET with the DOS command:

ARCNET 2<cr>

it means

(1) that this driver will find its relevant information in PORT2.CNF and

(2) you cannot load another RTES driver that will use PORT2.CNF

Your configuration file may contain the following parameters:

B=nnnn define an alternate base segment.
The default value is D000h or 53248 decimal.
If the arcnet 'nim' card on this node is jumpered for a different base segment, you may change the default value by including the B= parameter (the segment value must be in decimal)

C=x this parameter determines how your 'enable' register will operate. See chapter 13 of your RTES manual for details.

E=nnn defines the id of the first of a group of RTES register used for monitoring the network performance. See below. This register is optional. There is no default. If E is not defined, then the performance monitoring feature is switched off.

G=n defines the number of 'G' clusters to follow. You may define up to 512 'G' clusters.

I=n defines the number of 'I' clusters to follow. You may define up 512 'I' clusters.

P=nnn defines an alternate I/O port for the 'nim'. The default port is 2E0h or 736 decimal. If the 'nim' on this node is jumpered for a different I/O port, you may change the default value by including the P= parameter. (the port value must be in decimal.)

Q=n defines an alternate interrupt number for the 'nim'. The default interrupt is 4. If the 'nim' uses a different interrupt -IRQ- you may define the one use. IRQ 2 to 15 are supported.

T=nnn defines a communication time-out constant in milliseconds. The default value is 256.

Clusters are defined by 6 parameters:

FIRST, QTY, NODE, POS, ALERT, ENABLE

FIRST is the index of the first register in the group that will receive the values.

QTY is the number of registers to be transferred.
For I clusters, the maximum is 16 registers.
For G clusters, the maximum is 32 registers.

NODE is the node id of the 'nim' in the computer that the values will be coming from.

POS is the index of the first register at the source computer.

ALERT is the index of the register to be used for alerting of a communication time-out.

ENABLE is '0' for continuous communication or the index of the register that will be used to activate/deactivate this cluster. (See RTES user manual)

Monitoring network performance.

The E parameter defines the id of an RTES register. This Register is the first of a group of registers defined as follows :

********** LAYOUT OF ARCNET INTERFACE REGISTERS **********

Rnnnn	TOKEN	NO TOKEN
.	SHOW	MUST BE ON - THEN Ln - TO ACTIVATE BELOW
.	RI	RECEIVE INHIBIT
.	POR	POWER ON RESET
.	RECON	RECON FLAG
.	TA	TRANSMIT AVAILABLE
.	TATMO	TA TIME-OUT (MULTIPLE OF 3mS)
.	INPRO	# OF REQUESTS IN PROCESS
.	ROVFL	INPUT BUFFER OVERFLOW OCCURRENCE COUNTER
.	OBUFC	# OF BYTES IN OUTPUT BUFFER
.	DONE	# OF MESSAGES IN OUTPUT BUFFER
.	IBUFC	# OF BYTES IN INPUT BUFFER
.	PEND	# OF MESSAGES IN INPUT BUFFER

Note : E=nnnn-1

The first register 'TOKEN' indicates a 1 if the ARCNET card does not receive the token after a period of time of 15 mS. This situation may occur when the card is defective or when there is no other node on the network, or there is another node with the same ID.

The second register 'SHOW' tells the driver wether or not to update the other registers. The value of this register is read on startup, when you type the CONF command and when you type the Ln command. If SHOW is 0 (or any even number), then the other registers (RI to PEND) are not updated and you may use them for any other purpose. If SHOW is 1 (or any odd number), the RI to PEND registers are updated regularly by the ARCNET driver. (Names used are for illustration purposes. You may use any names you want for these registers, their function is not affected.)

Example configuration file.

The nim will use port1 : the parameters will be in PORT1.CNF

The nim will use the default segment and I/O port.
(B and P do need to be defined)

```
E=1024 ;register R1025 will annunciate network failures.
T=1000 ;time out constant (typical value for a AT 12 MHz)
G=3            ;will receive 3 groups of register values.
16,16,10,16,1,0 ;R0017 to R0031 will receive the values of ;R0017 to R0031 from node 10. R0002 will
        ;annunciate time outs. Communication will be ;continuous.
32,16,12,16,2,0 ;R0033 to R0048 will receive the values of ;R0017 to R0031 from node 12. R0003 will
        ;annunciate time outs. Communication will be ;continuous.
48,16,14,16,3,0 ;R0049 to R0064 will receive the values of ;R0017 to R0031 from node 14. R0004 will
        ;annunciate time outs. Communication will be ;continuous.
I=1            ;will receive 1 group of register status (0 or 1)
256,8,20,0,4,0  ;R0257 to R0272 will receive the status of ;R0001 to R0016 from node 20. R0005 will
        ;annunciate time outs. Communication will be ;continuous.
```

RTES - OPTOMUX NETWORK EMULATOR

This driver allows a system running RTES to be addressed by another system, via a serial communication port as if RTES were an OPTOMUX network or boards on an OPTOMUX network.

Serial port characteristics.

Just like an OPTOMUX network, OPTO_NET operates with

1 start bit
8 data bits
no parity
1 stop bit

You may use OPTO_NET in two different ways. You may let the system assign its own addresses, or you may impose which addresses will correspond to the RTES register pages.
1. Automatically assigned OPTOMUX addresses

To operate in this mode, make sure that PORTn.CNF does not contain anything except the B and M parameters.

In this mode of operation, each RTES page is automatically an OPTOMUX address.

The 16 'I' pages correspond to address 0 to 15
The 16 'O' pages correspond to address 16 to 31
The 16 'A' pages correspond to address 32 to 47

The 16 'Y' pages correspond to address 48 to 63
The 128 'R' pages correspond to address 64 to 191

The first variable on the page corresponds to OPTOMUX position 0.

2. User defined addressing.

In this method, you decide which OPTOMUX addresses will correspond to which RTES pages.

To define the addresses for the RTES pages, you include, for each page type, a definition line in PORTn.CNF that has the form:

t=n p,a p,a p,a ...

where t is the page type - I, O, A, Y, or G (for R pages)

n is the number of pages you want to use as OPTOMUX boards

each "p,a" couple is the page index and the OPTOMUX address that the page corresponds to. Note that the page indexes start at 0. For instance the page you would display with the command "I 5", has the index 4.

Notes:

Any RTES variable may be configured as input or output.

The I and O variables act as binary.

The A and Y variables act as analog, they transmit or receive their raw values, RTES converting from and to engineering units.

The R variables act as analog, they transmit or receive their integer portion from 0 to 4095.

OPTOMUX commands supported.

Since many of the OPTOMUX brain board functions are not applicable to RTES, OPTO_NET only handles a subset of the OPTOMUX command dictionary. It basically performs the data transfer functions.

These are the commands supported:

Command	Function	RTES response (if differ from OPTOMUX)
'A'	Power up	
'B'	Reset	
'C'	Set turn around delay	ACK but no action
'D'	Set watchdog delay	ACK but no action
'F'	Identify module type	
'G'	Configure positions	
'H'	Configure as inputs	
'I'	Configure as outputs	
'J'	Write analog outputs	
'K'	Force digital outputs on	
'L'	Analogs: Read inputs Digitals: Force outputs off	
'M'	Analogs: Read and average inputs Digitals: Read status	same as 'L'
'S'	Update analog outputs	

Any other command causes OPTO_NET to respond with a NAK.

NOTE REGARDING EXCHANGE OF DATA IN A DISTRIBUTED SYSTEM

When exchanging data between RTES nodes, the blocks of data are not transferred in any specific order. If your application must regard a set of data as one entity (a data base record, for instance), it is important to transmit the whole set within the same block, so that it is received as one block. This would eliminate the possibility of some of the information to be associated with other pieces of information that form part of the previously transmitted record. In particular, if some of the information is binary, but not all, it would be necessary to translate the binary information into a number 0 or 1 and make it part of the G or S block.

Example: A slave reads a temperature and sends it to the host, along with the time at which the temperature was read. The data consists of : temp, hr, min. If these 3 numbers are not send within the same block, RTES may see a new value of temp while hr and min are the ones for the previous value of temp.

USING EXTERNAL CODE WITH RTES

1. Introduction.

Although the rules and object oriented language (tasks) of RTES allow the execution of user defined algorithms, you may want to incorporate into RTES a user defined program. RTES is designed to run device drivers consisting of external programs that are memory resident. You may replace one or more of these drivers with your own program(s).

2. Requirements.

In order to make RTES run a user defined program, you must make sure that the program complies with certain criteria, as explained below.

A. Memory resident.

The user defined program must be memory resident and loaded into memory before you start up RTES. This is accomplished by the use the 'terminate and remain resident ' DOS directive.

B. Start address.

The start address of your program is composed of 2 sixteen bit words loaded into 2 consecutive memory locations. The 1st word is the offset corresponding to the start address of the program, the second word is the code segment where the offset is located. The start address of the user program must be placed into one of the following locations, normally used by device drivers. (all numeric data is in Hexadecimal)

```
0000:0180    offset   (Interrupt vector 60h)
0000:0182    cs

0000:0184    offset   (Interrupt vector 61h)
0000:0186    cs

0000:0188    offset   (Interrupt vector 62h)
0000:018A    cs

0000:0200    offset   (Interrupt vector 80h)
0000:0202    cs

0000:0204    offset (Interrupt vector 81h)
```

0000:0206 cs

The 1st set (180-182) is normally used by a PORT0 device driver such as PAMUX or RTC.

The 2nd set (184-186) is normally used by a PORT1 driver.

The 3rd set (188-18A) is normally used by a PORT2 driver.

The 4th set (200-202) is normally used by a PORT3 driver.

The 5th set (204-206) is normally used by a PORT4 driver.

RTES generates these interrupts on a regular basis so that the external code will get a chance to execute.

C. Integrity.

Your program must preserve all segment registers. It must not contain any long loops that will keep the processor unduly busy, since this will affect the performance of RTES.

3. Interface with RTES

Int 67H allows your external program to access some of the resources of RTES. When invoking INT 67H, the value of BP (base pointer) must contain the function number to be performed. The function numbers are :

```
0 (SETPT)     ;set the value of a variable
1 (GTVAL)     ;get the value of a variable
6 (ASTFP)     ;convert an ascii string to floating point
7 (FLOAT)     ;convert an integer to floating point
9 (FPTAS)     ;convert a floating point to ascii string
```

It is convenient to establish the following equates in your program, to improve readability:

```
SETPT EQU  0
GTVAL      EQU  1
ASTFP EQU  6
FLOAT      EQU  7
```

```
FPTASEQU  9
```

4. Point identification

RTES recognizes 5 types of variables. Within each type, the variables are numbered starting from zero (index). The 3 most significant bits identify which variable type is referred to:

Var. type	Mask (high byte)
Y	00100000B
A	01000000B
I	10000000B
O	10100000B
R	11000000B

For instance, the register R0006 is identified as C005h.

5. Floating Point register.

A 32 bit floating point register is reserved to communicate via INT 67H. You interface with it by using FLOAT, ASTFP or FPTAS. When converting to and from ASCII, you must use a buffer situated in the data segment that RTES uses when it enters your driver. The address of such buffer is located for your convenience at offset 0 of a table pointed to by SI. This buffer is 80 byte long.

This is how you access this buffer:

```
;DS is the original data segment
;Free DS for your own use.
        PUSH DS
        POP  ES
        .
        .
        .
;Get a buffer in the RTES data segment
        MOV BX,ES:[SI]
;Store character AL in this buffer
        MOV ES:[BX],AL
        INC BX
```

.
.

6. Using INT 67H

FLOAT AX must contain the integer

SETPT AX must contain the variable id and the floating point register must contain the new value.

GTVAL AX must contain the variable id. The call returns the current value in the floating point register and the corresponding integer in AX.

ASTFP The string must be pointed to by SI, in the original data segment (the one you have to preserve). The floating point value is placed in the floating point register.

FPTAS The floating point register must contain the floating point value. CX must contain the number of decimals (or 0 if you wish to use the default). You must supply a buffer pointed to by BX, in the original data segment, that will contain the ascii representation.

7. Example

Move the value of R0001 to R0002

```
mov   ax,0c000h        ;R1
mov   bp,GTVAL
int   67h              ;Put value of R1 in floating point register
mov   ax,0c001h        ;R2
mov   bp,SETPT
int   67h              ;Set R2 to the value in the floating point register
```

WRITING YOUR OWN EXTERNAL DEVICE DRIVER

The criteria and methods used for writing your own external device drivers are the same as those used for writing and using external code. With device drivers you must remember that the driver must set up the interrupt service routines for the ports being used by the device or poll the ports for data from the device. If polling the device, you should not keep the processor unduly busy since this will affect the performance of RTES. For example, your driver should detect when the device

is not responding so that it does not wait indefinitely.

Interrupts 63h, 64h, 82h and 83h are software interrupts initiated by the RTES clock tic. These interrupts are associated with the PORT1, PORT2, PORT3 and PORT4 drivers, respectively. These interrupts are normally used to perform the serial output to the appropriate COM port. When you write your own driver, you must provide at the corresponding locations, the start address of your serial output routine.

Characters coming in through the serial adapters are normally handled via an interrupt routine tied to the corresponding IRQ (interrupt request line). Your driver must provide such routine, as well as the interrupt handling mechanism.

CHAPTER SIXTEEN - HIGH RESOLUTION GRAPHICS

OVERVIEW

The high-resolution graphics generator allows you to prepare your own customised dynamic graphics displays. When a variable is on display, it is automatically updated to reflect the current status or value. As with the user display pages, variables can be displayed in a variety of formats.

While in high-resolution graphics mode, individual pixels can be addressed thus allowing x-y type graphs to be plotted. Combined with the scrolling function, RTES can be used to provide dynamic trend recording of single or multiple points on single or multiple graphs.

If you are using an EGA or VGA display, you may also use graphics displays that you created with other programs (PC Paintbrush, etc.) as the part of the display. Solid objects in the display can have their colors changed; variables and graphs can be displayed at strategic locations.

The high resolution graphics displays are used only to present information to the user. They do not allow data entry. While a graphics is on display, depressing any key (with a few exceptions) causes RTES to exit the high resolution graphics mode and return to whatever was on display previous to the display of the graphics. For example, when a high resolution graphics is on display RTES does not allow any commands to be entered. An exception to this is the <PrintScreen> key. If depressed, it will cause a dump of the screen to be sent to the printer.

You may use the high resolution graphics generator while RTES is operating. This means that you can modify or create high resolution graphics displays as the need arises, while your knowledge base is on line and while other tasks are running.

The high resolution graphics generator operates as an external memory resident driver which responds to commands from the RTES task executor. Each high resolution graphics display is generated by a task that contains the graphics commands. All commands that have to do with the creation, maintenance and control of tasks described in the section Tasks also apply to the high resolution graphics display tasks.

If the graphics driver is not installed, the task containing graphics commands will end when execution of a graphics command is attempted.

SCREEN RESOLUTION

When in graphics mode the screen has two coordinate systems - pixel coordinates and character coordinates.

Pixel Coordinates

When using the CGA graphics adapter cards the maximum line value is 199 and the maximum column value is 639. For example, the resolution is 200 by 640. The bottom left hand corner is line 0, column 0. Line 199 is the top line on the display. Column 639 is the right most column on the display. For example, the instruction

```
LINE 0 0 199 639 1 C
```

draws a diagonal line from the bottom left hand corner of the display to the top right hand corner of the display.

When using the EGA or VGA graphics adapter cards the screen resolution is determined by the video mode selected at the time the driver is loaded. The table below shows the video modes available for the standard EGA and VGA cards.

Video mode	Resolution Horizontal	Vertical	Adapter type
13	320	200	EGA,VGA
14	640	200	EGA,VGA
16	640	350	EGA,VGA
18	640	480	VGA

The maximum line value is one less than the resolution values. For example the maximum line value is 349 and the maximum column value is 639. when the resolution is 350 by 640.

When using the Hercules graphics adapter cards the maximum line value is 347 and the maximum column value is 719. For example, the resolution is 340 by 720.

Character Coordinates

When using the CGA, EGA or VGA graphics adapter cards, commands that use character coordinates have minimum line and column values of 0. Line 0 is the uppermost or top line of the display. Column 0 is the leftmost column of the display. The maximum line value depends on the number of pixel lines used for character. The maximum column value is 80. Column 80 is the right most line of the display area.

CGA characters are 8 pixel lines high. Therefore, the maximum character line is 24. HGA characters are 12 pixel lines high. Therefore, the maximum line value when using the Hercules graphics adapter cards is 28. EGA and VGA character height depends on the video mode or vertical resolution. The table below shows the lines per character for the various vertical resolutions available.

Vertical Resolution	Number of lines per character
200	8
350	12
480	16

Reserved Areas of the Screen

As with the customised display pages, certain portions of the screen are reserved for RTES system information. The reserved areas are the top (System title and time of day) line and bottom (Alert message) line. These areas are not useable by the graphics commands.

When using the CGA driver, character coordinate lines 0 and 24 are not usable. Usable pixel coordinate lines are 8 to 191.

When using the EGA driver, character coordinate lines 0 and 24 are not usable. Usable pixel coordinate lines are 14 to 335.

When using the HGA driver, character coordinate lines 0 and 28 are not usable. Usable pixel coordinate lines are 13 to 339.

GRAPHICS COMMANDS

The graphics commands are:

GRAPH	Enter graphics mode (and get a graphics display from disk if using the EGA driver
LINE	Draw a line
VAR	Display a variable dynamically
TEXT	Display a character string (static)
SCROL	Scroll a section of the screen to the left
COLOR	Change the color of a block of pixels

All other task command words can be used in tasks that generate a high resolution graphics display. They are described in full detail in the Tasks chapter.

The commands LINE, VAR, TEXT, SCROL and COLOR are only executed when the screen is in high resolution graphics mode. The command GRAPH causes the screen to enter high resolution graphics mode. Otherwise, they cause the task to terminate.

SYNTAX

The general syntax of graphics commands follows the same rules as that for the other task commands. For example, an instruction must start with an integer label that identifies the position of the instruction in the sequence; the next element must be a command word; then come the optional arguments. Spaces or commas must separate the elements of the instruction.

When the color argument is optional, a color of white is the default.

The formats of the graphics instructions are:

label GRAPH display_register<cr>
label LINE start_line start_column end_line end_column line_thickness color<cr>
label VAR line column variable specification<cr>
label TEXT line column color text<cr>
label SCROL top_line left_column number_of_lines number_of_columns<cr>
label COLOR line column color<cr>

GRAPH Command

The GRAPH command places the screen in high resolution graphics mode. If using the EGA driver, this command will read the graphics display file (PCX format, 640 by 350

resolution) pointed to by the display_register into the video buffer area.

LINE Command

The LINE command is used to draw a line between two specified pixels.

The start/end line/column values of the LINE command use the pixel coordinates. These values may be variables or constants.

If the line or column values are less than the minimum or greater than the maximum, RTES will only draw the line to the minimum or maximum values.

In the LINE command, the line_thickness value indicates how many line thicknesses the line will be. For example, the instruction

```
LINE 10 0 199 639 3 COLOR
```

will draw a diagonal line that is 3 times as thick as the diagonal line drawn by the instruction

```
LINE 10 0 199 639 1 COLOR
```

The same result can be generated by the following instructions:

```
LINE 10 0 199 639 1 COLOR
LINE 11 0 200 639 1 COLOR
LINE 12 0 202 639 1 COLOR
```

The line thickness value must be a constant.

The color value indicates the color that will be used to draw the line. When using monochrome drivers the color value if non-zero is interpreted as white as white. A zero value is interpreted as black. The color driver uses only the least significant 4 bits of the color value for drawing. Therefore, if the color value is greater than 16, the color used is the value of the remainder when the number is divided by 16.

The color value may be a variable or a constant.

The special feature of the LINE command is the characteristic that each time a line is redrawn, it is XORed with what is already there. For example, the instructions:

```
LINE 10 12 20 12 1 C
LINE 10 12 20 12 1 C
```

will first cause a line to be drawn on the display and then the line to be erased.

VAR Command

The VAR command is used to define a variable to be displayed dynamically on the screen at the defined line and column. The line and column specifications use the character coordinates. They must be constants.

The VAR command need only be executed once per variable after the GRAPH command.

The VAR variable specification follows the same format as that used in the Customised User Displays with the following exceptions:

1. Variables cannot be operator selectable.

2. The VGRAPH and HGRAPH specifications use the pixel coordinates for the total number of positions to be occupied by the full scale value and the maximum number that can be used for the positions is 255. In other words, the longest graph can only occupy 255 pixels wide or high.

The color parameter is used to specify both the foreground and background colors. The least significant 4 bits in the number are used to specify the foreground color. The next 4 bits are used to specify the background color. Therefore when you want to specify a foreground background combination you must multiply the background color by 16 and add the foreground color to it. For example, to get white on bright blue, the color value would have to be 151 (16*9+7).

TEXT Command

The TEXT command is used to display static text on the screen.

The line and column values for the TEXT command may be constants or variables. This command uses character coordinates.

The color parameter may also be a constant or a variable. As in the VAR command, the color parameter specifies both the foreground and background colors.

SCROL Command

The SCROL command is used to scroll a rectangular area of the screen one character to the left.

The parameters for the SCROL command may be constants or variables. This command uses character coordinates.

The number_of_lines and number_of_columns parameters specify the size of the block in characters to be scrolled one character to the left. A character occupies a block 8 pixels wide by 8 pixels high.

Example:

The following task will display the text of the variable LABEL in blue, draw a green box around it, and then continuously erase the box, and redraw the box.

```
1       GRAPH
2       VAR 10 30 LABEL TEXT LABEL 40 1
3       LINE 108 180 135 180 1 2
4       LINE 180 180 180 560 1 2
5       LINE 180 560 108 560 1 2
6       LINE 108 560 108 180 1 2
7       GOTO 3
```

COLOR COMMAND

The COLOR command is used to change the color of a block of pixels. A block of pixels is defined as a group of connected pixels of the same color. The line and column parameters specify the location of any pixel in the block. The color parameter specifies the new color for the block. The color of holes within the block will not be changed.

All parameters may be either constants or variables.

SCREEN DUMP

The non-system areas of the graphics display may be dumped to a dot matrix Epson FX-80 type printer or a HP LaserJet III printer by pressing <PrintScreen> (<PrtSc>). All non-black pixels are written to printer. Therefore, if you have a colored foreground on a colored background, you will not be able to distinguish between the two when the display is dumped to the printer.

The configuration file GPRINTER.CNF allows you to define which printer you have as your LPT1 device. You may also use the configuration file to define the character sequences used to output pixel graphics to your printer if you are not using one of the two supported types.

The **L** parameter defines which printer type you want to use. L=0 for Epson FX-80 type printers. L=1 for HP LaserJet III type printers. L=0 is the default setting. For example, if GPRINTER.CNF is not found, an Epson FX-80 type printer is assumed.

Other parameters may also be defined to allow the use of other types of printers capable of producing pixel graphics. The following parameters may be used:

R - append CR/LF to the end of output string
Format: R=0 or R=1 0 - yes, append CR/LF to end of string (default

for Epson FX-80)
1 - no, do not append CR/LF to end of string (default for HP LaserJet III)

C - convert binary number to ASCII string

Format: C=0 or C=1

0 - no, do not convert binary word to ASCII string. Use a binary number to indicate the number of data bytes being sent (default for Epson FX-80)

1 - yes, convert binary word to ASCII string. Use an ASCII string to indicate the number of data bytes being sent (default for HP LaserJet III)

S - start graphics mode sequence

Format: S=n $B_1...B_n$

n is the number of bytes in the setup sequence

$B_1...B_n$ are the decimal values of the ASCII bytes in the setup sequence

For example, the Epson FX-80 printer uses:

S=3 27,51,24

The HP LaserJet III uses:

S=17 27,69 27,38,108,49,79 27,42,114,48,70 27,42,114,48,65

E - end graphics mode sequence, return to text mode

Format: E=n $B_1...B_n$

n is the number of bytes in the end sequence

$B_1...B_n$ are the decimal values of the ASCII bytes in the end sequence

For example, the Epson FX-80 printer uses:

E=2 27,64

The HP LaserJet III uses:

E=6 27,42,114,66 27,69

H - data block header sequence prior to the number of data blocks to be sent

Format: H=n $B_1...B_n$

n is the number of bytes in the data block header sequence

$B_1...B_n$ are the decimal values of the ASCII bytes in the data block header sequence

For example, the Epson FX-80 printer uses:

E=2 27,75

The HP LaserJet III uses:

E=3 27,42,98

T - tail of data block header sequence after the number of data blocks to be sent

Format: T=n B_1...B_n n is the number of bytes in the tail of the data block header sequence

B_1...B_n are the decimal values of the ASCII bytes in the tail of the data block header sequence

For example, the Epson FX-80 printer does not use a tail.

The HP LaserJet III uses:

T=1 87

If these parameters are defined in the configuration file, they should follow the L parameter. In other words the L parameter must be the first parameter defined.

In addition to setting up the default parameters for the two printer types, the L parameter also defines how a data byte is to be interpreted by the printer. When L=0, a data byte defines 8 vertical pixels from the screen. In other words, one string of data bytes sent to the Epson FX-80 printer represents 8 rows of pixels from the screen.

When L=1, a data byte defines 8 horizontal pixels. Therefore, one string of data bytes sent to the HP LaserJet III printer represents one row of pixels from the screen.

The printer configuration parameters are loaded when the driver is installed. To change printer configuration parameters, the computer must be rebooted and the graphics driver re-installed.

INSTALLATION

The high resolution graphics generators are external memory resident drivers that must be loaded prior to RTES being loaded.

The color graphics driver to be used with VGA and EGA graphics cards is called EGA.EXE.

The monochrome graphics driver to be used with CGA, VGA and EGA graphics cards is called CGA.EXE.

The monochrome graphics driver to be used with Hercules graphics cards is called HGA.EXE.

To install a graphics driver type the name of the driver, without the .EXE extension in response to the DOS prompt.

For example:

EGA<cr>

will load the color graphics driver.

For the color graphics driver you may pass the video mode number as a parameter. By default video mode is 16 (640 by 350 16 color).

For example:

EGA 18<cr>

load the color graphics driver in 640 by 480, 16 color high resolution graphics mode.

EXAMPLE

Dynamic Trend Recording

The following example shows how to create a dynamic trend of a variable anywhere on the screen. The area to contain the trend is encircled by a box. The size and location of the area to be used can be determined by the user on the fly. The rules are used to scale the variable into the defined area. The task TREND does the actual trending. The task VAR is used to simulate a triangular waveform for the variable VAR. The task TREND also contains some examples of how to display variables in different formats on the graphics display page.

```
   NAME      CURRENT   A E  MESSAGE
R001 R001       0.00
R002 X0         5.00    TOP LEFT CHARACTER COORDINATE
R003 Y0         4.00    ============================
R004 W         30.00    WINDOW WIDTH
R005 H          3.00    WINDOW HEIGHT
R006 X          0.00    CURRENT X COORDINATE
R007 VAR       22.00    VARIABLE
R008 S8         0.00    COUNTER FOR SCROLL
R009 Y        274.00   R CURRENT Y COORDINATE (VARIABLE)
R010 L         38.00   R LEFT HAND BORDER
R011 R        281.00   R RIGHT HAND BORDER
R012 T        295.00   R TOP BORDER
R013 B        250.00   R BOTTOM BORDER
R014 0        252.00   R Y COORDINATE ORIGIN
R015 TY         2.00   R
R016 LPC       14.00   R
R017           0.00
R018           0.00
R019 Q         1.00   R GOING UP
```

```
R020                0.00    GOING DOWN

RULE Y=VAR+(25.00-Y0-H)*LPC
RULE L=8.00*(X0)-2.00
RULE R=8.00*(X0+W)+1.00
RULE T=LPC*(25.00-Y0)+1.00
RULE B=LPC*(25.00-Y0-H)-2.00
RULE 0=(25.00-Y0-H)*LPC
RULE TY=Y0-2.00
RULE LPC=14.00
RULE Q=(Q!VAR=(LPC*H-1))&(VAR>1.00)

LISTING OF TASK : TREND
001 GRAPH R001
002 TEXT TY X0  3 DYNAMIC TREND
003 LINE T L T R  2  3
004 LINE B L B R  2  3
005 LINE B L T L  2  3
006 LINE B R T R  2  3
007 VAR  2  25 VAR NUMBER  5 3
008 VAR  14  65 Q LIST  2 Q 12 44
009 VAR  20  75 VAR VGRAPH  100 100 4
012 LET X=(X0+W-1.00)*8.00
013 LET S8=8.00
014 WAIT 0.10
015 LINE Y X Y X  1  12
016 LET X=X+1.00
017 LET S8=S8-1.00
018 IF (S8>0.00) 015
019 SCROL Y0 X0 H W
020 GOTO 012

LISTING OF TASK : VAR
001 LET VAR=0.00
002 LET VAR=VAR+1.00
003 WAIT 0.10
005 IF (VAR<(LPC*H-1.00)) 002
010 LET VAR=VAR-1.00
011 WAIT 0.10
015 IF (VAR>0.00) 010
020 GOTO 002
```

GET SCREEN UTILITY

A get screen utility accompanies the high resolution graphics driver. This utility is a memory resident program that allows you to get the contents of the video buffer and

store it in a disk file in PCX format.

The program file name is GS.EXE. It must be installed in memory prior to your displaying a screen in graphics mode by entering GS in response to a DOS prompt.

GS is activated by pressing <Ctrl-Alt-UpArrow> when the graphics you want is being displayed. The current contents of the video buffer are stored in a file called SCREEN(n). For example, the first time GS is activated, it uses the file name SCREEN. The next time it uses the file name SCREEN1, and then SCREEN2, and so on.

GS remains in memory until the computer is rebooted.

CHAPTER SEVENTEEN
CONFIGURATION PARAMETERS

RESERVED VARIABLE NAMES

RTES recognizes the following variables on startup.

If RTES does not find one of these variable names the first time it should use it, it does not look for it any longer.

In order to use these registers, you must define them (see EDITING THE DATA BASE) then exit the RTES and reload it, so that the variables are seen upon program loading.

You may use these registers in rules, tasks, displays, reports, etc.

It is possible to overwrite the value of these registers using the SET command. This could be useful to test your knowledge base. However, the system will eventually set them back to the current values if applicable.

RTES does not necessarily have to see all the registers. It will use the ones it finds.

ACK$ RTES maintains this register to contain the quantity of acknowledged alerts.

ALARM$ RTES maintains this register to contain the quantity of currently active alerts (acknowledged and unacknowledged)

BL$ Defines text array for user help - bottom left of screen

BR$ Defines text array for user help - bottom right of screen

COS$ Defines disk file for change of state logging

CFn$ Defines array for <Ctrl-Fn> key (n=1 to 10)

DCOS$ Controls output of change of states to disk file

DISK_CL Reports the number of available disk cluster in the default drive or the drive

defined by the V parameter in RTES.CNF

DLOGS$ Controls output of alert events to disk file

Fn$ Defines array for <Fn> key (n=1 to 10)

HCOS$ Controls output of change of states to historian

INTERVAL Used by the trend recording facility

LOG$ Defines disk file for alert event logging

LABEL$ Defines text array for user page titles

LOGGING Used by the trend recording facility

PAGE$ Contains the display page number currently on the screen (1, 2, ..16). It contains 0 if the screen is not currently displaying a display page. PAGE$ will contain the display page number that was on the screen when you display an extended help file.

PCOS$ Controls output of change of states to printer

PLOGS$ Controls output of alert events to printer

PRINT$ Set to 0 if RTES is not currently printing a report, Set to 1 if RTES is currently printing a report.

READING$ Set to 0 if RTES is not currently executing a GET command. Set to 1 if RTES is executing a GET command.

SEL$ RTES will maintain it to show the identification number of the currently selected variable, using the following conventions.

0	no variable selected
00001 to 09999	register number
10001 to 19999	digital input number+10000
20001 to 29999	digital output number+20000
30001 to 39999	analog input number+30000
40001 to 49999	analog output number+40000

SYSPG$ maintained by RTES at a value that indicates what is being displayed. The value

of SYSPG$ is set by RTES whenever the screen changes display.

The values of SYSPG$ are as follows

1	Register definition pages (R,I,O)
2	Active alarms (LIST)
3	User display (PAGE)
4	Sheduler (SCHED)
5	A pages
6	Y pages
7	On line help (<Alt-H>)
8	Not used
9	List of display pages (PAGE<cr>)
10	Trend definition
11	Alarm historian (TRACE)
12	Not used
13	Alarm extension file (AEX) or extended help file (HLP)

RTES updates on start-up and every hour, on the hour, the following registers, if they are found when the program is loaded.

YEAR$ current year as returned by DOS (4 digits)

MONTH$ current month number as returned by DOS (Jan=1, etc.)

DAY$ current day as returned by DOS

DOW$ day of the week as returned by DOS (Monday=1, etc.)

HOUR$ current hour as returned by DOS (24 hour clock)

MINUTE$ current minute as returned by DOS - updated every minute

SECOND$ current second as returned by DOS - updated every second

See the appropriate chapters for details, in particular, Alert system, Trend Recording and Display Generator.

CONFIGURATION FILE

Some operating parameters for a RTES application are defined in the user modifiable ASCII text file RTES.CNF.

The configuration file is a text file created off-line and read by RTES on start-up.

Parameter definitions must be separated by a comma, space, linefeed, or carriage return.

Comment lines may be used. Lines beginning with a semi-colon are interpreted as comments. Comment lines are displayed at the bottom of the screen when the RTES.CNF file is read.

The parameters recognized are

A=n where n is the maximum number of analog input pages to display.

B=0 or 1 if 1 any variables using a variable as color attribute are continuously redisplayed. If 0, variables are only displayed when their value change (nit their color). Default is 0.

C=1 or 0 to indicate that you have or not (respectively) a numeric coprocessor and that you want RTES to use it. (default is 0)

D=1 or 0 to include or not (respectively) the date in the change of state messages. (default is 0)

E=n defines register index n as 'error' register to report DOS error number associated with the GET command.

F=1 or 0 to select or not (respectively) the automatic selection of the first variable on the screen. (default is 1)

H=1 or 0 to display or not display (respectively) the help line.(default is 1)

I=n where n is the maximum number of binary input pages to display.

K=n where n is the index of the first of 4 RTES 'R registers used for clock synchronization.

L=0,1,2,or 3 user level.

M=0 to 3 machine speed. (default is 0)

O=n where n is the maximum number of binary output pages to display.

P=1 to 1000 inference engine priority. (default is 100). This parameter should always be at its default value, unless you are using RTES for simulations that would make the inference engine particularly busy -- in which case you could use a lower value. Values higher than 100 are reserved for future applications.

R=n where n is the maximum number of 'R' register pages to display.

S=1 or 0 to use or not to use (respectively) the RTES internal printer control. (default is 1)

T=1 or 0 Normally, when RTES converts a value to an integer (NUMBER or LZI format), it changes any negative numbers to 0. The display range is from 0 64000.

You may change RTES to interpret integers as 2's complement by specifying in RTES.CNF the parameter T=1.

This causes any integer greater than 32767 to be interpreted as a negative number. The display range becomes -9999 to 32767. Note that the negative sign occupies one of the display positions.

This option also applies to numbers received via device drivers.

U=n to set up a block of registers starting at id n for mouse monitoring and control.

V=0 to n to indicate which drive will be used to return DISK_CL (see TREND RECORDING). V=0 is for drive A:, 1 for drive B:, etc. The default is the current drive.

Y=n where n is the maximum number of analog output pages to display.

USER LEVEL

There are 4 user levels. They are listed in the table below.

level	description
0	operator
1	operator with view
2	programmer
3	special functions

Each level defines the commands that the user has access to. Level 0 gives the user the ability to use the knowledge base but not to modify it. Level 1 gives the user the ability to use and view the knowledge base but not to modify it. Level 2 gives the user the ability to use all but the professional level commands. This level allows the user to define and maintain the knowledge base. Level 3 allows access to special commands (FTASK, FTEXT) only used to transfer a knowledge base between different versions of RTES.

The default user level is 0.

MACHINE SPEED

This parameter may be used to increase the throughput of your system for the serial ports and the print spooler - provided the compter you are using can handle it. The lowest speed factor is used with the slowest computer. The highest speed is used with the fastest computer. The higher the speed factor the higher the number of clock tics per second used by the scheduler. If the speed factor is too high, the response to keyboard inputs is noticeably reduced. The maximum speed factors that may be used with each computer are shown below. We recommand that you keep the M parameter at its default value, unless you have been advised otherwise in the documentation relative to the device drivers your are using.

The default machine speed is 0.

The M parameter can be located anywhere in the configuration file.

PRINTER CONTROL.

RTES normally uses LPT1 or LPT2 (whichever is found) and buffers its output to the printer, sending the characters provided the printer is not busy ('spooler'). This implies that it actually reads the printer port status.

In some applications, you may want to let your operating system - DOS or network - handle the printer control. For instance, you could have your printer on a serial port or at a file server. In this case, you may specify S=0 in RTES.CNF to tell RTES to pass on the printing to your operating system via the DOS interface mechanism.

EXAMPLE CONFIGURATION FILE

```
;Programmer Level
L=2
;User explanations will be defined by user
H=0
;End of configuration file
```

SYSTEM CONFIGURATION

Standard RTES

External data acquisition and control driver(s) : up to 5 devices

Quantity of variables (points)

Binary inputs	256
Binary outputs	256
Analog inputs	256
Analog outputs	256
Registers	2048

Data base size: 32,000 (words)

Maximum number of concurrent timers 64

Maximum number of active alerts 64

Maximum number of display pages 16

Maximum number of variables per page 96

Maximum number of scheduled events 10

Maximum number of tasks 8

Maximum number of statements per task 132

Note:

This configuration applies to this release only. It does not constitute system limits.

RTES is available with other configurations. Consult RTS for the configuration that meets with your requirements

SYSTEM CONFIGURATION

RTES-OEM

External data acquisition and control driver(s) : up to 5 devices

Quantity of variables (points)

Binary inputs	64
Binary outputs	64
Analog inputs	64
Analog outputs	64
Registers	512

Data base size: 10,000 (words)

Maximum number of concurrent timers 64

Maximum number of active alerts 64

Maximum number of display pages 16

Maximum number of variables per page 96

Maximum number of scheduled events 10

Maximum number of tasks 8

Maximum number of statements per task 132

Note:

This configuration applies to this release only. It does not constitute system limits.

RTES is available with other configurations. Consult RTS for the configuration that meets with

(c)1992-2005 FAI Designs Inc.

your requirements

SYSTEM CONFIGURATION

RTES-MAX (Jumbo)

External data acquisition and control driver(s) : up to 5 devices

Quantity of variables (points)

Binary inputs	1024
Binary outputs	1024
Analog inputs	512
Analog outputs	512
Registers	3072

Data base size: 64,000 (words)

Maximum number of concurrent timers 64

Maximum number of active alerts 64

Maximum number of display pages 16

Maximum number of variables per page 96

Maximum number of scheduled events 10

Maximum number of tasks 16

Maximum number of statements per task 132

Note:

This configuration applies to this release only. It does not constitute system limits.

RTES is available with other configurations. Consult RTS for the configuration that meets with your requirements

CHAPTER EIGHTEEN - RTES COMMANDS

DISPLAY AN ANALOG INPUT DEFINITION PAGE **A**

Format: A n<cr>

Parameter: n page number.

Default: If the page number is omitted, or too large, the first page is displayed.

Note: This command allows the subsequent use of <PgDn> and <PgUp> and of the EDIT command.

Example: A 3<cr>

Access Level: 1

Reference: 5-2, 5-4

ACKNOWLEDGE THE MOST RECENT/SELECTED ALERT **ACK**

Format: ACK<cr> or <F10> (Acknowledge the selected alert or most recent alert)
ACK name<cr> (Acknowledge the alert 'name')
ACK line<cr> (Acknpwledge the alert on line 'line')
ACK @reg<cr> (Acknowledge the alert on the line indicated by the value of 'reg'.

Parameters:
- name the name of a RTES variable
- line a constant integer between 4 and 20
- reg the name of a RTES 'R' variable

Access Level: 0

Reference: 12-2

DISPLAY THE ALERT EXTENSION FILE FOR THE MOST RECENT/SELECTED ALERT **AEX**

Format: AEX<cr>
or <Alt-X> (Display the file for the selected alert or most recent alert)
AEX name<cr> (Display the file for the alert 'name')
AEX line<cr> (Display the file for the alert on line 'line')
AEX @reg<cr> (Display the file for the alert on the line indicated by the value of 'reg'.

Parameters:

	name	the name of a RTES variable
	line	a constant integer between 4 and 20
	reg	the name of a RTES 'R' variable

Access Level: 0

Reference: 12-6

Notes: Alerts are selectable on the active alerts page (LIST) or the alert/change of state historian page (TRACE).

This command is for applications that use the mouse and the EXEC function to execute commands normally entered via the keyboard.

DEFINE AN ALERT STATE **ALERT**

Format: ALERT var s<cr>

Parameters:

	var	variable name, other than analog.

s 0 alert on FALSE state
1 alert on TRUE state
E change of state logging
N no alert state.

Example: ALERT HIGH 1<cr>

An alert will be initiated if HIGH is TRUE.

ALERT HIGH N<cr>

The previously defined alert condition for HIGH is cancelled.

Access Level: 2

Reference: 5-10 12-2

CALCULATE THE ARC COSINE **ARC**

Format: ARC angle cos<cr>

Parameters: angle register that will receive the angle in radians

cos register that contains the cosine

Example: ARC ARCOS RATIO<cr>

Whenever RATIO changes, its arc cosine will be placed in ARCOS.

Access Level: 2

Reference: 7-6

CREATE A CUSTOMIZED DISPLAY PAGE **BUILD**

Format: BUILD name<cr>

Parameter: nameA unique alphanumeric with a maximum of 10 characters from the set A-Z, 0-9, @,#, $,%,_,?.

Example: BUILD INFO<cr>

A new display page named INFO is created and open for editing.

Access Level: 2

Reference: 9-1

CLEAR SCREEN **CLS**

Format: CLS<cr>

The active portion of the screen is erased. This command is mainly used in conjunction with the Ln and M monitoring commands, to view new streams of data more comfortably.

Access Level: 1

FILL AN AREA OF THE SCREEN **COLOR**

Format: COLOR line col color<cr>

Parameters: line the pixel line of a dot in the area to be filled

col the pixel column of a dot in the area to be filled

color the color to fill the area with

The screen must be in pixel graphics mode. All three parameters may be constants or variable names. The area filled is the one enclosing the dot specified with the same color as the dot.

Note: This command cannot be executed from the keyboard. It may be part of a task or a batch file or it may be an EXEC statement.

Access Level: n/a

Reference: 16-7

REDO THE CONFIGURATION **CONF**

Format: CONF<cr>

Cancels any parameters defined in the previous RTES.CNF and PORTn.CNF. Implements the configuration defined by the new contents of the *.CNF files. Initializes all the device drivers.

Note: This may cause a communication error if the command is issued while a driver was receiving data from the device. Unless the communication fault registers are re-defined, the error would clear by itself.

Access Level: 3

COPY A FILE **COPY**

Format: COPY source dest<cr>

Parameters: source full path name of the file to be copied

dest full path name of the destination file

Copies the source file to the destination file. If the destination file already exists, it is deleted and a new one created.

Both source and destination may be indirectly specified using file array registers.

Example: COPY EVENTS A:EVENTS<cr>

The file called EVENTs in the default directory is copied to drive A.

COPY @EVENTS @BACKUPS<cr>

The path name specified by the array EVENTS is copied to the path name specified by the array BACKUPS.

Access Level: 0

CREATE A TEXT FROM A TEMPLATE FILE **CTEXT**

Format: CTEXT FILE FILE.FMT

Uses FILE.FMT to produce a text and assign it to register FILE.

FILE.FMT may contain conditionals and variables using the report generator notation.

Access Level: 3

SET THE DATE **DATE**

Format: DATE mm dd yyyy<cr>

Parameters: mmmonth of the year (January is month 01)

dd day of the month

yyyy year

Example: DATE 07 15 1993<cr>

Access Level: 0

Note: This command sets the DOS date. Depending on your version of DOS, this command may or may not be passed on to your hardware real time clock.

SET ON/OFF THE DEBUG MODE **DEBUG**

Format: DEBUG<cr> or <alt-D>

Toggles on and off the DEBUG mode. When DEBUG is on, using the RULE command to display an existing rule, causes all the variables in the rule to be displayed with the

function key variable control enabled. The word DEBUG appears at the top of the screen.

The DEBUG feature also expands the other functions (PID, etc.)

Access Level: 1

Reference: 6-11

SET THE NUMBER OF DIGITS AFTER THE DECIMAL POINT **DEC**

Format: DEC n<cr>

Parameter: n an integer between 0 and 8.

This number of decimals applies to all real numbers specified with a default field width and number of decimals, in displays and reports.

Example: DEC 3<cr>

All real numbers without field specification will show 3 decimals.

Access Level: 1

Reference: 9-6

DECREMENT THE SELECTED VARIABLE **DECR**

Format: DECR<cr>

Parameter: none

This command is for applications that use the mouse and the EXEC function to execute commands normally entered via the keyboard.

Access Level: 0

Reference:

DELETE A FILE **DEL**

Format: DEL file_name<cr>

Parameter: file_name the path name to be deleted.

Deletes the specified file. The file may be indirectly specified using a file array register.

Example: DEL ALARMS<cr>

The file 'ALARMS' in the default directory is deleted.

DEL @ARCHIVE<cr>

The path name defined by the file array ARCHIVE is deleted.

Access Level: 0

OPEN/CLOSE DISK FILE TO SUBSTITUTE FOR PRINTER **DOC**

Format: DOC name<cr>

Parameter: name name of disk file to act as printer output.

Default: If 'name' is not specified, this commands closes the current file (if any) that was opened by a previous 'DOC name' command.

The file may be indirectly specified using a file array register.

Example: DOC REG.TXT

The file REG.TXT in the default directory will substitute for the printer

until a DOC command is issued.

DOC @FILE

The path name specified by the file array FILE will substitute for the printer until a DOC command is issued.

Access Level: 3

ENTER THE EDIT MODE **EDIT**

Format: EDIT<cr> or <alt-E>

This command is valid only if the current display was not obtained by the I,O,R,A,Y, or PAGE command. All other displays are 'write protected'.

Access Level: 2

Reference: 5-7 9-2

TERMINATE AN ACTIVE TASK **END**

Format: END name<cr>

Parameter: namename of the task to be terminated.

Example: END AVRGE<cr>

Access Level: 1

Reference: 8-9

DELETE A TASK FROM THE SYSTEM **ERASE**

Format: ERASE name<cr>

Parameter: namename of the task to be removed.

Example: ERASE RAMP<cr>

Access Level: 2

Reference: 8-11

ERASE ERROR MESSAGE **ERMS**

Format: ERMS<cr>

Parameter: none

This command is for applications that use the mouse and the EXEC function to execute commands normally entered via the keyboard. It erases any error message displayed by the system on line 23 of the screen.

DEFINE ENGINEERING UNIT **EU**

Format: EU name text<cr>

Parameters: namename of the analog point.

text engineering unit consisting of up to 3 characters.

Example: EU TACH RPM<cr>

Access Level: 2

Reference: 5-10

DEFINE A COMMAND LIST **EXEC**

Format: EXEC name=expression<cr>

Parameters: name name of the register which is used as

index to a command list.

expression — expression that determines the value of the index register.

The value of the index determines which command in the list is to be executed each time the value of the expression changes.

Example: EXEC INDEX=1*(T<1)+2*(T>0&T<100)+3*(T>99)<cr>

Access Level: 2

Reference: 7-5

DEFINE AN EXTERNAL CODE TRIGGER **EXT**

Format: EXT name=expression<cr>

Parameters: name — name of the register to be associated with the external memory resident code pointed to by interrupt vector 65H.

expression — expression that defines the value of the above register.

Whenever the value of the expression is affected the external code is called. The point id and value of the register are passed to the external code.

Example: EXT EXT_CALC=RPM>6500<cr>

Access Level: 2

Reference: 7-6

CREATE A COMMAND BATCH FILE **FILE**

Format: FILE file_name<cr>

Parameter: file_name path name of the destination file.

Creates a file containing a series of SET and TEXT commands affecting the variables listed in a 'format' file named DATA.FMT. The format file must be in the default directory.

The destination file may be specified indirectly using a file array register.

Example: FILE SETUP<cr>

The file SETUP will be created in the default directory.

FILE @CASE

A file specified by the array CASE will be created.

Access Level: 0

Reference: 13-3

DUMP A DISPLAY PAGE TO DISK **FPD**

Format: FPD<cr>

Dump the current user display page in compressed format to a disk file. The file name is composed of the display name with the extension '.PIC'

Access Level: 3

ERASE A RULE OR FUNCTION **FREE**

Format: FREE name<cr> or <alt-F>name<cr>

Parameter: name name of the variable evaluated by a rule or special function to be removed.

Example: FREE TOTAL<cr>

Any function (RULE, PID, etc.) associated with TOTAL is erased from the knowledge base.

Access Level: 2

Reference: 6-4 7-9

LOAD A TASK FROM A DISK FILE **FTASK**

Format: FTASK name<cr>

Parameter: name Disk file containing the task listing. First word in file is the name of the task to be created.

Example: FTASK RAMP.TXT

The task in the text file RAMP.TXT is added to the system.

Access Level: 3

Reference: 8-15

LOAD A VARIABLE DEFINITION FILE **FTEXT**

Format: FTEXT name<cr>

Parameter: name Disk file containing the variable definition. This file must have been created by directing to disk - using 'DOC name' - the output of PI, PO, PA, PY or PR.

Example: FTEXT VARS.TXT

Access Level: 3

EXECUTE A BATCH FILE **GET**

Format: GET name<cr>

Parameter: name the path name of the file to be executed.

The batch file may be specified indirectly using a file array register.

Example: GET TEST.CMD<cr>

GET @RECIPE

Access Level: 0

Reference: 13-1

SUSPEND EXECUTION OF A TASK **HALT**

Format: HALT name<cr> or <F6>name<cr>

Parameter: name name of the task to be suspended.

Example: HALT RAMP<cr>

Access Level: 1

Reference: 8-8

DISPLAY CONTEXT SENSITIVE HELP FILE **HLP**

Format: HLP<cr>

Display the user defined context sensitive help file associated with the currently selected variable.

Note: The HLP command can only be executed from a batch file, a task or an EXEC statement.

<Esc> performs the same function while a user page is displayed.

Access Level: n/a

Reference: 9-19

RETURN TO COMMAND MODE **HOME**

Format: HOME<cr>

Ends an editing session. Cancels the selection of a variable on the screen.

Note: The HOME command can only be executed from a batch file, a task or an EXEC statement.

<Home> performs the same function.

Access Level: n/a

Reference: 9-13

DISPLAY A DIGITAL INPUT DEFINITION PAGE **I**

Format: I n<cr>

Parameter: n page number.

Default: If omitted or too large, the first page is displayed.

Note: This command allows the subsequent use of <PgDn> and <PgUp> and of the EDIT command.

Example: I 3<cr>

Access Level: 1

Reference: 5-2

GET THE RTES SERIAL NUMBER **ID**

Format: ID<cr>

The serial number is displayed at the bottom of the screen.

Access Level: 0

INCREMENT THE SELECTED VARIABLE **INCR**

Format: INCR<cr>

Parameter: none

Note: This command is for applications that use the mouse and the EXEC function to execute commands normally entered via the keyboard.

Access Level: 0

Reference:

TURN ON/OFF THE COMMUNICATION PROTOCOL MONITOR FOR PORT0 **L0**

Format: L0<cr>

Toggles on/off the monitoring.

Access Level: 0

Reference: 15-25

Note: Drivers associated with PORT0 do not usually display the exchange of data between the computer and the device (if any). This commands is provided to make any driver on PORT0 to identify itself.

TURN ON/OFF THE COMMUNICATION PROTOCOL MONITOR FOR PORT1 **L1**

Format: L1<cr>

Toggles on/off the monitoring.

Access Level: 0

Reference: 15-25

TURN ON/OFF THE COMMUNICATION PROTOCOL MONITOR FOR PORT2 **L2**

Format: L2<cr>

Toggles on/off the monitoring.

Access Level: 0

Reference: 15-25

TURN ON/OFF THE COMMUNICATION PROTOCOL MONITOR FOR PORT3 **L3**

Format: L3<cr>

Toggles on/off the monitoring

Access Level: 0

Reference: 15-25

TURN ON/OFF THE COMMUNICATION PROTOCOL MONITOR FOR PORT4 **L4**

Format: L4<cr>

Toggles on/off the monitoring.

Access Level: 0

Reference: 15-25

ASSIGN A CALCULATED VALUE TO A VARIABLE **LET**

Format: LET name=expression<cr>

Parameters: name name of the variable to be calculated.

expression a boolean/arithmetic expression.

Example: LET RATE=(A+B)/C<cr>

Access Level: 0

Note: This is different from a RULE, since it does not create a permanent relationship between 'name' and 'expression'.

Reference: 5-13

DRAW A LINE IN PIXEL MODE **LINE**

Format: LINE y0 x0 y1 x1 thick color

Parameters: y0,x0 coordinates of the start point. (Constants or variables)

y1,x1	coordinates of the end point. (Constants or variables)	
thick	thickness of the line. (Constant)	
	color	color attribute. (Constant or variable)

Note: The screen must be in pixel. This command may only be executed by a task, a batch file or an EXEC statement.

Access Level: n/a

Reference: 16-2 16-4

DISPLAY THE CURRENT ACTIVE ALERTS **LIST**

Format: LIST<cr> or <alt-L>

This command allows the subsequent use of <PgDn> and <PgUp>.

Access Level: 0

Reference: 12-3

LOCATE A VARIABLE BY NAME **LOC**

Format: LOC name<cr>

Parameter: name name of the variable to be looked up.

RTES finds the variable and displays the page where it is located. A * flashes next to the variable name.

Access Level: 1

DEFINE A LOOP **LOOP**

Format: LOOP cv fb sp trig<cr>

Parameters: cvvariable to be used as the control variable for the loop

fb variable to be used as the feedback variable for the loop

sp variable to be used as the set-point variable for the loop

trig an optional variable used to force the loop evaluation

This command defines a loop with control variable cv, feedback fb and set-point sp. This command assigns the 5 registers immediately after sp to become respectively: P coefficient, I coefficient, D coefficient, previous error and auto/manual switch.

Default: If only cv is entered in the command line, the other 2 parameters are displayed if the LOOP exists.

Example: LOOP VALVE FLOW NOM_FLOW<cr>

Access Level: 2

Reference: 7-1

TURN ON/OFF THE INFERENCE ENGINE MONITOR **M**

Format: M<cr>

Toggles on/off the display of the activities of the inference engine. While the monitoring is on, the screen shows the names of the variables as they are evaluated by the inference engine.

SELECTED THE VARIABLE POINTED TO BY THE MOUSE **MS**

Format: MS<cr>

Parameter: none

This command is for applications that use the mouse and the EXEC function to execute commands normally entered via the keyboard.

Access Level: 0

Reference:

ASSIGN A NAME TO A VARIABLE **NAME**

Format: NAME Xn name<cr>

Parameters: X I,O,R,A or Y

n variable number

name name to be assigned to the variable. Must contain only letters and digits, start with a letter and have a maximum of 10 characters from the set A-Z, 0-9, @,#, $,%,_,?.

Example: NAME R10 TOTAL<cr>

(Assign the name TOTAL to the register #10)

Access Level: 2

Reference: 5-9

CREATE/EDIT A TEXT FILE **NOTE**

Format NOTE name<cr>

Parameter: name the path name of the disk file to be created or edited.

Example: NOTE MEMO.TXT<cr>

Note: This feature allows the editing of text files that are 15 lines or less. If you call up a longer file and press <Esc>, only the first 15 lines will be saved - the rest will be lost.

Access Level: 0

DISPLAY A DIGITAL OUTPUT DEFINITION PAGE **O**

Format: O n<cr>

Parameter: n page number.

Default: If n is omitted or too large, the first page is displayed.

This command allows the subsequent use of <PgDn> and <PgUp> and of the EDIT command.

Example: O 2<cr>

Access Level: 1

Reference: 5-2

OPEN A TASK FOR EDITING **OPEN**

Format: OPEN name<cr> or <Alt-O>name<cr>

Parameter: name name of task to be viewed or edited.

Example: OPEN RAMP<cr>

Access Level: 2

Reference: 8-3

PRINT THE ANALOG INPUT DEFINITIONS **PA**

Format: PA<cr>

Access Level: 1

Reference: 14-3

DISPLAY THE NAMES OF THE CURRENT DISPLAY PAGES **PAGE**

Format: PAGE<cr> or <alt-P><cr>

Error conditions: None.

Access Level: 0

Reference: 9-14

DISPLAY A CUSTOMIZED PAGE **PAGE**

Format: PAGE name<cr> or <alt-P> name<cr>

Parameter: name the name of the page to be displayed.

Default: If name is omitted, the list of all available pages is displayed. You may then use the <Tab> or <Arrow Dn> key to select a name and press <cr> to obtain the display.

This command allows the subsequent use of <PgDn> and <PgUp> and of the EDIT command.

Example: PAGE INFO<cr>

Access Level: 0

Reference: 9-2 9-14

PRINT ALERT HISTORY **PAH**

Format: PAH<cr>

The contents of the 'TRACE' buffer is printed.

Access Level: 0

Reference: 12-5

DUMP A DISPLAY PAGE TO DISK **PD**

Format: PD<cr>

Dump the current user dispay page in compressed format to a disk file.
The file name is composed of the display name with the extension '.PIC'

Access Level: 3

Reference: 9-18

INSTALL PRINTER DEFINITION FILE **PDEF**

Format: PDEF name<cr>

Parameter: name name of the printer definition file (.PDF extension is assumed)

Example: PDEF EPSON<cr>

Access Level: 0

Reference: 14-4

PRINT THE KNOWLEDGE BASE **PE**

Format: PE<cr>

The RULEs and other functions are printed.

Access Level: 1

Reference: 14-3

NEXT PAGE **PGDN**

Format: PGDN<cr>

Same as <PgDn>

Access Level: 0

PREVIOUS PAGE **PGUP**

Format: PGUP<cr>

Same as <PgUp>

Access Level: 0

PRINT THE DIGITAL INPUT DEFINITIONS **PI**

Format: PI<cr>

Access Level: 1

Reference: 14-3

DEFINE A VELOCITY TYPE PID LOOP **PID**

Format: PID cv fb sp trig<cr>

Parameters: cv variable to be used as the control variable for the loop

fb variable to be used as the feedback variable for the loop

sp variable to be used as the set-point variable for the loop

trig an optional variable used to force the loop evaluation

Defines a pid loop with control variable cv, feedback fb and set-point sp. This command assigns the 6 registers immediately after sp to become respectively : P coefficient, I coefficient, D coefficient, previous error, auto/manual switch and error at time-2.

Default: If only cv is entered in the command line, the other 2 parameters are displayed if the PID exists.

Example: PID VALVE FLOW NOM_FLOW<cr>

Access Level: 2

Reference: 7-2

LOAD A DISPLAY PAGE FROM DISK **PL**

Format: PL name<cr>

Parameter: name name of the (.PIC) file containing a previously dumped display page.

Example: PL FORM<cr>

Access Level: 3

Reference: 9-18

PRINT THE DIGITAL OUTPUT DEFINITIONS **PO**

Format: PO<cr>

Access Level: 1

Reference: 14-3

PRINT THE REGISTER DEFINITIONS **PR**

Format: PR<cr>

Access Level: 1

Reference: 14-3

PRINT A REPORT **PRINT**

Format: PRINT report_name first last<cr>

Parameters: report_name disk file to be used by the report generator

first first page number to be printed

last last page number to be printed

Default: If first and last are not specified, the complete report is printed.

Example: PRINT A:PROGRESS.REP<cr>

PRINT PRODUCT.REP 3 6<cr>

Access Level: 0

Reference: 14-1

PRINT THE SCREEN **PRTSC**

Format: PRTSC<cr> or <PrtSc>

Lines 4 to 20 of the screen are printed.

<PrtSc> is also available in pixel mode. The complete screen except the bottom text line is printed.

Access Level: 0

Reference: 3-2

PRINT ALL SCHEDULED EVENTS **PS**

Format: PS<cr>

Access Level: 1

Reference: 10-2

PRINT A TASK LISTING **PT**

Format: PT name<cr>

Parameter: name name of the task to be printed.

Example: PT RAMP<cr>

Access Level: 1

Reference: 8-14

PULSE AN OUTPUT OR REGISTER **PULSE**

Format: PULSE name time<cr> or <F4>name time<cr>

Parameters: name the name of the output or register to pulse

time duration of the pulse in seconds. (default is 0.8 seconds)

Example: PULSE LIGHT 3<cr>

Access Level: 0

Reference: 5-11

PRINT THE ANALOG OUTPUT DEFINITIONS **PY**

Format: PY<cr>

Access Level: 1

Reference: 14-3

DISPLAY A REGISTER DEFINITION PAGE **R**

Format: R n<cr>

Parameter: n page number.

Default: If n is omitted or too large, the first page is displayed.

Note: This command allows the subsequent use of

<PgDn> and <PgUp> and of the EDIT command.

Example: R 2<cr>

Access Level: 1

Reference: 5-2

DEFINE ANALOG INPUT RANGE **RANGE**

Format: RANGE name zero full<cr>

Parameters: name name of analog input point.

zero value in engineering units when the analog input raw value is 0.

full full scale value in engineering units.

Example: RANGE TEMP -40 300<cr>

Access Level: 2

Reference: 5-10

RENAME A TASK **REN**

Format: REN old new<cr>

Parameters: old current name of the task.

new new name of the task

Example: REN RAMP STEP<cr>

Access Level: 2

Reference: 8-11

DEFINE/RECALL A RULE **RULE**

Format: RULE name=exp<cr> or <Alt-R>name=exp<cr>

Parameters: name variable to be evaluated by the rule.

exp a boolean/arithmetic expression.

Default: If only 'name' is specified, the rule associated with that variable is displayed.

Example: RULE TOTAL=A+B+C<cr>

Access Level: 2

Reference: Chapter 6

INITIATE OR RESUME EXECUTION OF A TASK **RUN**

Format: RUN name<cr> or <F5>name<cr>

Parameter: name name of the task to start or resume execution.

Example: RUN RAMP<cr>

Access Level: 1

Reference: 8-8

SAVE REVISED FILES ON DISK **SAVE**

Format: SAVE<cr>

Note that when you exit the Real-Time Expert System using <Ctrl-Home>, any revised OLEX.* files are automatically written back to the disk. This command is useful

during development as there is the possibility of a power failure that would abnormally shut down your computer (and loose your work).

Access Level: 2

DEFINE/RECALL A SCHEDULED EVENT **SCHED**

Format: SCHED name hh:mm action<cr> or <Alt-S>name hh:mm action<cr>

Parameters: name Name of the variable (point) to be controlled.

hh:mm Time of the event on 24hr clock.

action ON to start
OFF to stop
REMOVE to cancel this event

Default: If 'name' only is specified, any scheduled events associated with this variable are displayed.

Example: SCHED COFFEE 10:15 ON<cr>

Access Level: 2

Reference: 10-1

SELECT A VARIABLE OR AN ALART LINE **SEL**

Format: SEL<cr> selects the next variable or alert

SEL name<cr> selects the variable or alert 'name'

SEL line<cr> selects the alert at the screen line specified

SEL @reg<cr> selects the alert at the screen line specified by the value of the variable

Parameters: name name of the variable.
line a number between 4 and 21
reg the name of a RTES register

Example: SEL SET_POINT<cr>

Access Level: 0

Reference:

SET THE VALUE OF A VARIABLE TO A CONSTANT **SET**

Format: SET name value<cr> or <F3>name value<cr>

Parameters: name name of the variable.

value desired constant.

Example: SET TOTAL 38.6<cr>

Access Level: 0

Reference: 5-11

DEFINE/RECALL A CASCADE OF REGISTERS **SHIFT**

Format: SHIFT reg_name=exp<cr>

Parameters: reg_name the name of the register that causes the cascade to operate.

exp a boolean expression using the same syntax as a rule.

The two registers immediately following reg_name (in the list of registers) are assumed to contain the register number of the first register in the cascade and the quantity of

registers to cascade, respectively. On transition from false to true of reg_name, the value of each register in the cascade is transferred to the next register.

Default: If 'name' only is entered, the rest of the line is recalled and displayed.

Example: SHIFT TIME=TICK

Access Level: 2

Reference: 7-3

DETERMINE THE MEMORY USED AND AVAILABLE **SIZE**

Format: SIZE<cr>

The right hand side of the command line displays the number of words used by the tasks and rules and the number of words still available for tasks and rules.

Access Level: 2

SET A VARIABLE TO 1 OR ON **START**

Format : START name<cr> or <F1>name<cr>

Parameter: name the name of the variable to be controlled.

Example: START PUMP<cr>

Access Level: 0

Reference: 5-11

MAKE A TASK EXECUTE THE NEXT INSTRUCTION **STEP**

Format: STEP task_name<cr>

Parameter: task_name the name of the task

The task executes one statement and suspends itself.

Example: STEP AVERAGE<cr>

Access Level: 2

Note: The task must be inactive or suspended.

Reference: 8-10

SET A VARIABLE TO 0 OR OFF **STOP**

Format : STOP name<cr> or <F2>name<cr>

Parameter: name the name of the variable to be controlled.

Example: STOP PUMP<cr>

Access Level: 0

Reference: 5-11

ABORT PRINT **STP**

Format : STP<cr>

Access Level: 0

Note: RTES ends the current print jobs and start the next one (if any).

Any text that was already sent to the printer will continue printing.

Reference: 14-6

DISPLAY THE STATUS OF THE TASKS **TASK**

Format: TASK<cr> or <Alt-T>

Access Level: 2

Reference: 8-11

CREATE A TASK AND OPEN IT FOR EDITING **TASK**

Format: TASK name<cr>

Parameter: name name to be assigned to the task. May be up to 10 characters from the set A-Z, 0-9, @, #, $, %, _, ?.

Example: TASK CALC_AVGE<cr>

Access Level: 2

Reference: 8-2

ASSIGN A TEXT DESCRIPTOR TO A VARIABLE **TEXT**

Format: TEXT name text<cr> or <F9>name text<cr>

Parameters: name variable name to be assigned the text.

text up to 40 characters

Example: TEXT T_ALARM The temperature is too high<cr>

Access Level: 2

Reference: 5-9

COPY A TEXT FROM ANOTHER VARIABLE **TEXT**

Format: TEXT dest @@source<cr> or <F9>dest @@source<cr>

Parameters: dest destination variable name

source source variable name

Example: TEXT EVENTS @@B<cr>

Access Level: 2

Reference: 5-9

SET THE TIME OF DAY **TIME**

Format: TIME hh mm<cr>

Parameters: hh hour of the day (24 hour clock)

mm minute of the hour

Note: This command sets the DOS time. Depending on your version of DOS, this command may or may not be passed on to your hardware real time clock.

Example: TIME 12 05<cr>

Access Level: 0

DISPLAY ALERT AND CHANGE OF STATE HISTORY **TRACE**

Format: TRACE<cr>

This command allows the subsequent use of <PgDn> and <PgUp>

Access Level: 0

Reference: 12-5

DISPLAY THE TREND TABLE **TREND**

Format: TREND<cr>

Access Level: 1

Reference: 11-1

DISPLAY THE STATUS OF THE TASKS **TS**

Format: TS

Access Level: 0

Note: This is the same command as 'TASK<cr>'

Reference: 8-11

DISPLAY A TEXT IS PIXEL MODE **TXT**

Format: TXT line col color text<cr>

Parameters:	line the character line where to text will appear. (Constante or variable)
col	the character column where the text will start. (Constante or variable)
color	the color attribute. (Constante or variable)

	text	the text to display.

Note: The screen must be in pixel mode. Equivalent to the TEXT task directive. This command may only be used by a task, batch file or EXEC.

Example: TXT 3 42 12 TANK STATUS<cr>

TXT YT XT BRIGHT BATCH MONITORING<cr>

Access Level: n/a

Reference: 16-6

DEFINE ANALOG INPUT CONVERSION TYPE **TYPE**

Format: TYPE name x<cr>

Parameters:	name	name of analog input point
x	conversion type	

Example: TYPE TEMP J<cr>

Access Level: 2

Reference: 5-11

DISPLAY A VARIABLE IN PIXEL MODE **VAR**

Format: VAR line col name spec color<cr>

Parameters:	line	character line where the variable will appear. (Constante or variable)
col	character column of leftmost or lowest position. (Constante or variable)	

name the name of the variable to be displayed.

spec display specification.

color color attribute. (Constante or variable)

The variable is dynamically updated.

Note: The screen must be in pixel mode. Equivalent to the VAR task directive. This command may only be used by a task, batch file or EXEC.

Access Level: n/a

Reference: 16-5

CREATE A BATCH FILE USING A SPECIFIC FORMAT FILE **XFILE**

Format: XFILE f_file c_file<cr>

Parameters: f_file name containing the format file. (f_file.FMT)

c_file name of the destination file.

Creates a file containing a series of SET and TEXT commands affecting the variables listed in the 'format' file.

You may specify the destination file indirectly using a file array register

Example: XFILE PRODUCT TOM1<cr>

XFILE PRODUCT @TYPES<cr>

Access Level: 0

Reference: 13-3

DISPLAY A DISK RESIDENT PAGE **XP**

Format: XP name<cr>

Parameter: name name of file previously created with a PD or FPD command (name.PIC)

Example: XP EVENTS<cr>

Access Level: 0

Reference: 9-18

DISPLAY AN ANALOG OUTPUT DEFINITION PAGE **Y**

Format: Y n<cr>

Parameter: n page number.

Default: If n is omitted or too large, the first page is displayed.

Note: This command allows the subsequent use of <PgDn> and <PgUp> and of the EDIT command.

Example: Y 3<cr>

Access Level: 1

Reference: 5-2

RTES APPLICATION EXAMPLE

HOT WATER CONTROL

RTES monitors the hot and cold water sources to dispense the required flow at the required temperature. Two loop controls are used to set the hot and cold water valves according to the demand.

If the temperature is not within prescribed limits, flow is interrupted until the mix has reached the proper temperature.

The system allows tells when conditions cannot be reached.

The simulations capabilities of RTES allow to observe the operation of the system without the use of any equipment. Rules are used to simulate hot and cold water mixing to produce a given water temperature.

DATA BASE LAYOUT

```
R0097  HOT          L       HOT WATER TAP
R0098  COLD         L       COLD WATER TAP
R0099  HF           R       HOT WATER FLOW
R0100  CF           R       COLD WATER FLOW
R0101  FLOW         R       TOTAL FLOW
R0102  HTEMP                HOT WATER TEMPERATURE
R0103  CTEMP                COLD WATER TEMPERATURE
R0104  TEMP         R       MIX TEMPERATURE
R0105  FMAX                 MAXIMUM FLOW PER SOURCE
R0106  DTEMP                DESIRED TEMP
R0107  DFLOW                DESIRED FLOW
R0108
R0109  CC           R       COLORS
R0110  HC           R
R0111  MC           R
R0112
R0113  HF_SP        R       SET POINT FOR HOT
R0114  HP           R
R0115
R0116
R0117
R0118  HM           R
R0119
```

(c)1992-2005 FAI Designs Inc.

```
R0120
R0121  CF_SP         R       SET POINT FOR COLD
R0122  CP           R
R0123
R0124
R0125
R0126  CM           R
R0127
R0128
R0129  RUN          R
R0130  ADJ          R       ADJUSTING
R0131  TX           R       TEMPERATURE CANNOT BE ACHIEVED
R0132  FX           R       FLOW CANNOT BE ACHIEVED
R0133
R0134  DX           R
R0135
R0136
R0137
R0138
R0139
```

KNOWLEDGE BASE - (RULES)

```
;LOOP CONTROL - HM AND CM PROHIBIT CONTROL BEYOND THE OPERATING ;RANGE
LOOP HOT HF HF_SP
RULE HF_SP=((DTEMP-CTEMP)*DFLOW)/(HTEMP-CTEMP)
RULE HP=0.05
RULE HM=(HF_SP<0.00)!(HF_SP<FMAX)'&(HOT<FMAX)'
LOOP COLD CF CF_SP
RULE CF_SP=(DFLOW-HF_SP)
RULE CP=0.05
RULE CM=(CF_SP<0.00)!(CF_SP<FMAX)'&(COLD<FMAX)'
;FEED BACK IS SIMULATED BY MAKING FLOWS EQUAL TO VALVES
RULE HF=HOT*(DTEMP>CTEMP)
RULE CF=COLD*(DTEMP<HTEMP)
RULE FLOW=HF+CF
;THE RESULTING TEMPERATURE MEASUREMENT IS SIMULATED
RULE TEMP=(HTEMP*HF+CTEMP*CF)/FLOW
;TO SHOW WATER FLOW IN COLOR WHEN IT EXISTS
RULE CC=3.00*(CF>.5)
RULE HC=4.00*(HF>.5)
RULE MC=12.00*(FLOW>.50)*RUN
;WATER FLOWS ONLY IF TEMPERATURE IS CLOSE TO DESIRED
RULE RUN=TEMP>(DTEMP-0.50)&TEMP<(DTEMP+0.50)
;DIAGNOSTIC. IF WATER DOES NOT RUN, WE WANT TO KNOW WHY.
RULE ADJ=RUN'&(TX!FX)'
```

```
RULE TX=DTEMP<CTEMP!DTEMP>HTEMP
RULE FX=FLOW<DFLOW&((HF<FMAX)'!(CF<FMAX)')
RULE DX=ADJ+2.00*TX+3.00*(FX&TX')

;INITIAL CONDITIONS
SET HTEMP 100
SET CTEMP 20
SET FMAX 120
SET DFLOW 70
SET DTEMP 50
```

You may now change the desired conditions DFLOW and DTEMP and watch how the system reacts, adjusting the valves for hot and cold.

RTES APPLICATION

HEAT PUMP CONTROL

RTES REPLACES A CONVENTIONAL CONTROL SYSTEM TO PROVIDE MORE POWERFUL STRATEGIES.

In this example, in addition to switching between heating and cooling modes automatically, it also uses outside air when possible to save energy.

DATA BASE LAYOUT

```
R0065  ROOM_T
R0066  MIN_T
R0067  MAX_T
R0068  OA_T
R0069  HEATER        R
R0070  DAMPER        R
R0071  COMPRESSOR    R
R0072  FAN           R
R0073  DS                \2---
R0074                    \3///
R0075
R0076
R0077
R0078
R0079
R0080
```

KNOWLEDGE BASE (RULES)

```
RULE HEATER=(HEATER!ROOM_T<MIN_T)&(ROOM_T<MAX_T)&(OA_T>ROOM_T)'
RULE DAMPER=ROOM_T>MAX_T&COMPRESSOR'!ROOM_T<MIN_T&HEATER'
RULE COMPRESSOR=(COMPRESSOR!ROOM_T>MAX_T)&(ROOM_T>MIN_T)&(OA_T<ROOM_T)'
RULE FAN=DAMPER!COMPRESSOR!HEATER
```

(c)1992-2005 FAI Designs Inc.

```
;INITIAL CONDITIONS
SET MIN_T 18
SET MAX_T 23
SET OA_T 17
SET ROOM_T 20
```

RTES APPLICATION EXAMPLE

FIRE ALARM AUXILIARY SYSTEM

RTES ATTACHES TO A CONVENTIONAL F/A SYSTEM AND THE HVAC CONTROLS TO INTEGRATE THE FUNCTIONS AND GREATLY EXPANDS THEIR CAPABILITIES.

1. Smoke Control
RTES decide which air handling units, smoke dampers and fire doors releases must operate in order to contain the smoke by creating a negative pressure in the area(s) where smoke is detected.

2. Graphic representation
RTES displays dynamically the situation as it develops to aid in assessing the evacuation strategy.

3. Information
Alarm extension files provide complete information on how to reach the area in alarm and pertinent data as to unusual situations such as flammable materials in storage, special equipment required, etc.

4. Logging
Events are logged chronologically to help analyze what happened and spot deficiencies in the system.

5. Ease of update
As the building is modified (walls added or removed), new devices are installed or the use of the facilities change, the on-line editing allows to keep the Knowledge Base current.

DATA BASE LAYOUT

```
;Binary inputs: smoke detection units.

I0001  Z1S      1       SMOKE DETECTION ZONE 1
I0002  Z2S      1       SMOKE DETECTION ZONE 2
I0003  Z3S      1       SMOKE DETECTION ZONE 3
I0004  Z4S      1       SMOKE DETECTION ZONE 4
```

(c)1992-2005 FAI Designs Inc.

```
I0005  Z5S       1          SMOKE DETECTION ZONE 5
I0006  Z6S       1          SMOKE DETECTION ZONE 6
I0007  Z7S       1          SMOKE DETECTION ZONE 7
I0008  Z8S       1          SMOKE DETECTION ZONE 8
I0009  Z9S       1          SMOKE DETECTION ZONE 9
I0010  Z10S      1          SMOKE DETECTION ZONE 10
I0011  Z11S      1          SMOKE DETECTION ZONE 11
I0012  Z12S      1          SMOKE DETECTION ZONE 12
I0013
I0014
I0015
I0016



;Binary outputs : Air handling units, smoke dampers, door releases
O0001  AH1          R       Air handling unit#1
O0002  AH2          R
O0003  AH3          R
O0004  AH4          R
O0005  AH5          R
O0006  AH6          R
O0007  AH7          R
O0008  AH8          R
O0009  AH9          R
O0010  AH10         R
O0011  AH11         R
O0012  AH12         R       Air handling unit#12
O0014
O0015
O0016
O0017  SD1          R       Smoke damper
O0018  SD2          R
O0019  SD3          R
O0020  SD4          R
O0021  SD5          R
O0022  SD6          R
O0023  SD7          R
O0024  SD8          R
O0025  SD9          R
O0026  SD10         R
O0027  SD11         R
O0028  SD12         R
O0029  DR56_H       R       Door release zone 5 to 6
O0030  DR78_H       R       Door release zone 7 to 8
O0031
O0032
```

```
;Registers
;Registers are in lieu of analog inputs to simulate easily temperature
;changes.
```

```
R0017  Z1_T                Temp. in zone 1
R0018  Z2_T
R0019  Z3_T
R0020  Z4_T
R0021  Z5_T
R0022  Z6_T
R0023  Z7_T
R0024  Z8_T
R0025  Z9_T
R0026  Z10_T
R0027  Z11_T
R0028  Z12_T
R0029
R0030
R0031
R0032
;Patterns to be used to display an air handling unit, smoke damper, smoke ;condition.
R0033  AH                   \4+
R0034                       \2AH
R0035  SD                   \2---------
R0036                       \3/////////
R0037  SMOKE
R0038                       \4SMOKE
R0039
;To introduce a delay on system startup and prevent dampers from
;operating
R0040  POWER          R
R0041
R0042
R0043
R0044
R0045
R0046
R0047
R0048
```

KNOWLEDGE BASE (RULES)

```
RULE AH1=Z1S'&((AH1!(Z1_T>75.00))&(Z1_T>70.00)!Z2S!Z5S)
RULE AH2=Z2S'&((AH2!(Z2_T>75.00))&(Z2_T>70.00)!Z1S!Z3S!Z6S)
RULE AH3=Z3S'&(Z2S!Z4S!Z7S!(AH3!(Z3_T>75.00))&(Z3_T>70.00))
RULE AH4=Z4S'&((AH4!(Z4_T>75.00))&(Z4_T>70.00)!Z3S!Z8S)
RULE AH5=Z5S'&((AH5!(Z5_T>75.00))&(Z5_T>70.00)!Z1S!Z6S!Z9S)
RULE AH6=Z6S'&((AH6!(Z6_T>75.00))&(Z6_T>70.00)!Z2S!Z5S!Z7S!Z10S)
RULE AH7=Z7S'&((AH7!(Z7_T>75.00))&(Z7_T>70.00)!Z3S!Z6S!Z8S!Z11S)
RULE AH8=Z8S'&((AH8!(Z8_T>75.00))&(Z8_T>70.00)!Z4S!Z7S!Z12S)
RULE AH9=Z9S'&((AH9!(Z9_T>75.00))&(Z9_T>70.00)!Z5S!Z10S)
```

```
RULE AH10=Z10S'&((AH10!(Z10_T>75.00))&(Z10_T>70.00)!Z6S!Z9S!Z11S)
RULE AH11=Z11S'&((AH11!(Z11_T>75.00))&(Z11_T>70.00)!Z7S!Z10S!Z12S)
RULE AH12=Z12S'&((AH12!(Z12_T>75.00))&(Z12_T>70.00)!Z8S!Z11S)
RULE POWER=POWER'd5.00
RULE DR56=Z5S!Z6S
RULE DR78=Z7S!Z8S
RULE SD1=POWER&(Z1S'd5.00)'
RULE SD2=POWER&(Z2S'd5.00)'
RULE SD3=POWER&(Z3S'd5.00)'
RULE SD4=POWER&(Z4S'd5.00)'
RULE SD5=POWER&(Z5S'd5.00)'
RULE SD6=POWER&(Z6S'd5.00)'
RULE SD7=POWER&(Z7S'd5.00)'
RULE SD8=POWER&(Z8S'd5.00)'
RULE SD9=POWER&(Z9S'd5.00)'
RULE SD10=POWER&(Z10S'd5.00)'
RULE SD11=POWER&(Z11S'd5.00)'
RULE SD12=POWER&(Z12S'd5.00)'

;INITIAL VALUES
SET Z1_T 70
SET Z2_T 70
SET Z3_T 70
SET Z4_T 70
SET Z5_T 70
SET Z6_T 70
SET Z7_T 70
SET Z8_T 70
SET Z9_T 70
SET Z10_T 70
SET Z11_T 70
SET Z12_T 70
```

RTES APPLICATION EXAMPLE

BATCH HEATING

Upon signal from the push button START, an on/off valve FILL opens and the liquid fills the tank until the level indicator LEVEL reaches MAX. FILL closes, HEAT starts and heats up the liquid until TEMP reaches TMAX. Heating stops, and the system waits until the liquids cools off to DTEMP. A that time, a proportional valve VALVE ramps from 0 to 100% in 16.4 seconds and the tank empties. Once LEVEL reaches MIN, the proportional valve closes from 100 to 0% in 8.2 seconds. The system waits for the START signal to repeat the sequence. At any time, pressing ABORT causes everything to shut down. The array MS is used to display STATUS so that the user is informed of what is taking place.

Only registers are used in this example, to make simulation easier. Real I/O's would be associated with the registers in the actual application.

NAME	CURRENT	A	E	MESSAGE
R209 START	0.00			
R210 FILL	0.00	R		
R211 HEAT	0.00	R		
R212 HOT	0.00	R		
R213 EMPTY	0.00	R		
R214 VALVE	0.00	R		
R215 DV	0.00	R		
R216 DVX	0.00	R		
R217 P	0.10	R		
R218 TMAX	120.00	R		
R219 DTEMP	100.00	R		
R220 MIN	0.00			
R221 MAX	60.00	R		
R222 ABORT	0.00			
R223 CLOSE	0.00	R		
R224 STATUS	0.00	R		
R225 TEMP	0.00			
R226 LEVEL	0.00			
R227 INC	1.22	R		
R228 DEC	2.44	R		
R229 MS	0.00			PRESS START TO FILL THE TANK
R230 MS1	0.00			FILLING
R231 MS2	0.00			HEATING

(c)1992-2005 FAI Designs Inc.

R232 MS3 0.00 COOLING
R233 MS4 0.00 EMPTYING
R234 MS5 0.00 CLOSING VALVE
R235 0.00
R236 0.00
R237 0.00
R238 0.00
R239 0.00
R240 0.00

KNOWLEDGE BASE

RULE FILL=(START!FILL)&(LEVEL<MAX)&ABORT'
RULE HEAT=(LEVELgteMAX!HEAT)&(TEMP<TMAX)&HOT'&ABORT'
RULE HOT=(HOT!(TEMP<TMAX)')&(LEVEL>MIN)&ABORT'
RULE EMPTY=(EMPTY!HOT&TEMP<DTEMP)&(LEVEL>MIN)&ABORT'
RULE VALVE=VALVE+EMPTY*DV*INC*(VALVE<100.00)-CLOSE*DV*DEC
RULE DV=DVX'dP&(EMPTY!CLOSE)
RULE DVX=DVdP
RULE P=0.10
RULE TMAX=120.00
RULE DTEMP=100.00
RULE MAX=60.00
RULE CLOSE=LEVELlteMIN&(VALVE>0.00)&ABORT'
RULE STATUS=FILL+2.00*HEAT+3.00*(HOT&EMPTY')+4.00*EMPTY+5.00*CLOSE
RULE INC=100.00/(5.00*16.40)
RULE DEC=100.00/(5.00*8.20)

;ALTERNATELY, THE FOLLOWING TASKS WOULD PERFORM THE SAME FUNCTIONS
AND MAY REPLACE THE RULES : FILL, HEAT, EMPTY, CLOSE

LISTING OF TASK : PRO
001 IF [START'] 001
002 START FILL
003 IF [LEVEL<MAX] 003
004 STOP FILL
005 START HEAT
006 IF [TEMP<TMAX] 006
007 STOP HEAT
008 IF [TEMP>DTEMP] 008
009 START EMPTY
010 IF [LEVEL>MIN] 010
011 STOP EMPTY
012 START CLOSE
013 IF [VALVE>0.00] 013

```
014 STOP CLOSE
015 STOP START
016 GOTO 001
;
```

```
LISTING OF TASK : ABORT
001 IF [ABORT'] 001
002 X END PRO
003 STOP ABORT
004 STOP FILL
005 STOP HEAT
006 STOP EMPTY
007 STOP CLOSE
008 SET VALVE 0.00
009 STOP START
010 X RUN PRO
011 GOTO 001
```

RTES APPLICATION EXAMPLE

STATISTICAL QUALITY CONTROL

Each of three production machines provide the gauge readings Mn_X, Mn_Y and Mn_Z and signal via a digital input (Mn) that the data is available.

RTES collects the data in a separate file for each machine, the data is in spreadsheet format for further off-line analysis.

A task continuously looks at the last 10 readings from each machine (sliding window), calculates the average on those 10 readings, compares those averages to set limits and flags a machine as required.

The flag may be used to alert operators, stop production and/or adjust the machine appropriately.

DATA BASE LAYOUT

NAME	CURRENT	A	E	MESSAGE
I001 M1	OPEN			
I002 M2	OPEN			
I003 M3	OPEN			
I004	OPEN			
I005	OPEN			
I006	OPEN			
I007	OPEN			
I008	OPEN			
I009	OPEN			
I010	OPEN			
I011	OPEN			
I012	OPEN			
I013	OPEN			
I014	OPEN			
I015	OPEN			
I016	OPEN			

(c)1992-2005 FAI Designs Inc.

```
    NAME        CURRENT    A E  MESSAGE

R017 M1_X            0.00
R018 M1_Y            0.00
R019 M1_Z            0.00
R020                 0.00
R021                 0.00
R022 M2_X            0.00
R023 M2_Y            0.00
R024 M2_Z            0.00
R025                 0.00
R026                 0.00
R027 M3_X            0.00
R028 M3_Y            0.00
R029 M3_Z            0.00
R030                 0.00
R031                 0.00
R032                 0.00

    NAME        CURRENT    A E  MESSAGE

R033 RMC#              0.00
R034 WMC#              0.00     M1.DAT
R035 R0035             0.00     M2.DAT
R036 R0036             0.00     M3.DAT
R037                 0.00
R038                 0.00
R039                 0.00
R040 READ_POS          0.00
R041 SIZE             0.00
R042 TX               0.00
R043 TY               0.00
R044 TZ               0.00
R045 X                0.00
R046 Y                0.00
R047 Z                0.00
R048                 0.00
    NAME        CURRENT    A E  MESSAGE

R049 FLAGS             0.00
R050 MC#1F             0.00
R051 MC#2F             0.00
R052 MC#3F             0.00
R053                 0.00
R054                 0.00
R055                 0.00
```

```
R056            0.00
R057            0.00
R058            0.00
R059            0.00
R060            0.00
R061            0.00
R062            0.00
R063            0.00
R064            0.00

LISTING OF TASK : RECORD
001 OPEN   1 WMC#   3
002 IF [(M1!M2!M3)'] 002
003 IF [M1] 010
004 IF [M2] 020
005 IF [M3] 030
009 GOTO 002
010 SET WMC# 0.00
011 APP   1
012 WRITE   1 M1_X
013 IF [M1] 013
014 GOTO 002
020 SET WMC# 1.00
021 APP   1
022 WRITE   1 M2_X
023 IF [M2] 023
024 GOTO 002
030 SET WMC# 2.00
031 APP   1
032 WRITE   1 M3_X
033 IF [M3] 033
034 GOTO 002


LISTING OF TASK : SQC
001 OPEN   2 RMC#   3
002 LET RMC#=3.00
010 APP   2
011 GPOS   2 READ_POS
012 IF [READ_POS<10.00] 060
013 LET TX=0.00
014 LET TY=0.00
015 LET TZ=0.00
020 LET READ_POS=READ_POS-10.00
021 LET SIZE=10.00
025 POS   2 READ_POS
027 READ   2 X
```

```
030 LET TX=TX+X
031 LET TY=TY+Y
032 LET TZ=TZ+Z
035 LET SIZE=SIZE-1.00
036 IF [SIZE>0.00] 027
040 LET TX=TX/10.00
041 LET TY=TY/10.00
042 LET TZ=TZ/10.00
045 LET FLAGSiRMC#=(TX<12.00!TY>25.00!TZ>13.50)
060 LET RMC#=RMC#-1.00
061 IF [RMC#>0.00] 010
062 WAIT 10.00
063 GOTO 002
```

RTES Application example

DATA AVERAGING

This example is a step by step instruction to develop a system that maintains a running average of a variable over a time window.

We are going to define a set of 11 registers (A0 - A10) that will keep the current value and the 10 most recent values of the variable to be averaged. Three registers (TRIG, F, L) are required to operate the 'shift'. Another three registers (T, TX, D) will make a 'flip-flop' to generate a trigger at an adjustable time period. A register (AV) will contain the running average. A task - that we will name FAKE - will simulate that the variables is changing to generate some data. In an actual situation, the variable will be linked to actual analog inputs using rules.

We assume that you have started RTES.

1. Define the variables

To display a page of registers (the second page), type:

R 2<cr>

To edit the registers, type:

EDIT<cr>

The white patch is in front of R0017. Type:

TRIG<cr>
F<cr>
L<cr>
A0<cr>
A1<cr>
A2<cr>
A3<cr>
A4<cr>
A5<cr>
A6<cr>
A7<cr>
A8<cr>

(c)1992-2005 FAI Designs Inc.

A9<cr>
A10<cr>

Press <Pg Dn> to go to the next page of registers. The white patch will be at R0033. Type:

AV<cr>
T<cr>
TX<cr>
D<cr>

Press <home> to exit the editing.

2. Define the flip flop.

Type:

RULE D=1<cr>
RULE T=TX'd(D/2)<cr>
RULE TX=Td(D/2)<cr>

You will now see T go to 1 the 0 once per second.
You may change the period by giving another value to D
(i.e RULE D=10<cr>)

3. Define the shift register.

Type :

RULE F=20<cr>
RULE L=11<cr>

This establishes where the shift registers start and how many there are.

SHIFT TRIG=T<cr>

This makes the register TRIG a trigger for the shift registers defined by the next 2 variables (F and L).

4. Make AV the arithmetic average of A1 to A10.

Type :

RULE AV=(A1+A2+A3+A4+A5+A6+A7+A8+A9+A10)/10<cr>

5. The FAKE task.

To define a new task, type:

TASK FAKE<cr>

Now type the task:

1 SET A0 100<cr>
2 WAIT .2<cr>
3 LET A0=A0-0.5<cr>
4 IF [A0>0] 2<cr>
5 GOTO 1<cr>

Press <Home> to exit the task editing.

6. Run the FAKE task

To start the FAKE task, type:

RUN FAKE<cr>

7. Looking at the results

The <Pg Up> and <Pg Dn> keys will allow you to flip between the register pages you have used.

You will notice that as the primary variable A0 is changed by FAKE, the previous values are pushed down the next ten registers every time TRIG changes from 0 to 1.

Simultaneously, AV reflects the average of the past 10 values.

RTES Application example

A DATA LOGGER

This example is a step by step instruction to develop a small data logger.

We are going to define a set of 3 variables that will be recorded in a disk file at a regular adjustable interval. The recording will be done by a task that we will name LOG. A second task - that we will name FAKE - will simulate that one of the variables is changing to generate some data. In an actual situation, the set of variables may be linked to analog inputs using rules.

The file produced can be imported into spreadsheet programs.

We assume that you have started RTES.

1. Define the variables

To display a page of registers, type:

R<CR>

To edit the registers, type:

EDIT<CR>

The white patch is in front of R0001. Type:

V1<cr>
V2<cr>
V3<cr>

Press the <arrow down> 4 times to leave some room, then type:

FILE<cr>
NREC<cr>
DELAY<cr>

Press the <arrow up> 3 times to bring the patch to FILE, then press the <Tab> key twice to move to the text area, then type the file name :

(c)1992-2005 FAI Designs Inc.

DATA.PRN<cr>

(DATA.PRN must be on the same line as FILE).

Press <home> to exit the editing.

2. The LOG task

To define a new task, type:

TASK LOG<cr>

Now type the task:

1 OPEN 1 FILE 3<cr>
2 WAIT DELAY<cr>
3 WRITE 1 V1<cr>
4 GPOS 1 NREC<cr>
5 IF [NREC<100] 2<cr>
6 END<cr>

Press <Home> to exit the task editing.

3. The FAKE task.

To define a new task, type:

TASK FAKE<cr>

Now type the task:

1 SET V1 100<cr>
2 WAIT .2<cr>
3 LET V1=V1-0.5<cr>
4 IF [V1>0] 2<cr>
5 GOTO 1<cr>

Press <Home> to exit the task editing.

4. Define DELAY

To log the data every second, type:

RULE DELAY=1

5. Run the logger

To start the FAKE task, type:

RUN FAKE<cr>

To start the LOG task, type:

RUN LOG<cr>

The logging will end after 100 recordings, or as soon as you type:

END LOG<cr>

6. Looking at the results

To exit from RTES, press <Ctrl-Home>

You may now view the file DATA.PRN with the DOS command:

TYPE DATA.PRN<cr>

7. Notes

This example simply illustrates the RTES capabilities to produce a working application in a few minutes. You may enhance or modify this example to suit your own requirements.

The data may be replayed in RTES with a task that reads the file with a READ instead of a WRITE directive. You may view the data graphically using the RTES-PIX option. The data block may be longer, you may use different file names, etc.

REAL TIME EXPERT SYSTEM

PROCEDURE TO TRANSPORT A KNOWLEDGE BASE TO A DIFFERENT VERSION OF RTES

INTRODUCTION.

RTES stores the knowledge base you create in a series of files named

OLEX.PAG	(user defined display pages)
OLEX.NAM	(variable names)
OLEX.DAT	(rules, texts and tasks)
OLEX.LNK	(cross reference used by the inference engine)

These files are created and updated automatically by RTES in your current working directory.

These files are structured specifically for the version of RTES on which you produce them. This is necessary in order to optimize the memory used and the execution speed.

If you attempt execution of a knowledge base by a version of RTES that is not compatible, the message CORRUPT DATA BASE appears on the screen.

It is however possible to make any knowledge base compatible with any version of RTES. The procedure uses the RTES that originally created the knowledge base and a word processor.

Note that this same procedure may be used to 'clean up' an old knowledge base, if you wish to eliminate a large amount of variables and rules that you had previously defined, or rearrange the variables.

The procedure consists of dumping the knowledge base on disk files in text format, using the original RTES that created it, editing those text files and re-loading them into the new RTES that will be using them. The procedure is fully mechanized to insure an accurate transfer of everything in your knowledge base, with the exception of the alert conditions and the conversion parameters for the analog variables. These must be re-entered as if they had never existed.

(c)1992-2005 FAI Designs Inc.

RTS Inc. will be pleased to perform this procedure for you. Simply mail your diskette containing the OLEX files and state the version numbers of the original and target RTES. RTS Inc. will mail the diskette back to you, usually on the same business day, with the files fully converted. The fee for this service is $100.00, including handling and shipping.

The following describe this procedure step by step, if you wish to do it by yourself.

BACKUP

As for any modification procedure, it is a good idea to save a copy of the original OLEX files on a diskette, just in case you need to redo your procedure or consult what you had in the original knowledge base.

To do so, place a diskette in drive A: and type the DOS command

```
COPY OLEX.* A:<cr>
```

DUMPING THE OLEX FILES IN TEXT FORMAT ON DISK

You dump the knowledge base by using the 'DOC name' command a open the file, the appropriate print command to write to the file and the 'DOC' command to close the file.

The variables, the rules and each task must be sent to a separate file. The detailed procedure follows:

1. Start the original RTES with its knowledge base.

2. If you did not define a register called PRINT$, do so then exit RTES with <ctrl-Home> and restart it.

3. Make sure that all variables that have a text in the MESSAGE column has a name, otherwise the message will not be recoverable when loading into the new system. If R0045 has a text but no name, you may simply give it the name R0045.

4. Execute the following RTES commands

```
DOC V.TXT<cr>       ;open a file called V.TXT on the current directory
PI<cr>              ;write the binary inputs
```

PO<cr> ;write the binary outputs
PR<cr> ;write the registers
PA<cr> ;write the analog inputs
PY<cr> ;write the analog outputs

(You may of course skip any variable type that you are not using)

5. Wait for PRINT$ to switch to 0, indicating that the system has finished writing to the disk.

6. Execute the RTES command

DOC<cr> ;close the V.TXT file

7. Execute the following RTES commands

DOC E.TXT<cr> ;open a file called E.TXT on the current directory
PE<cr> ;write the rules, loops, cascade registers.

8. Wait for PRINT$ to switch to 0, indicating that the system has finished writing to the disk.

9. Execute the RTES command

DOC<cr> ;close the E.TXT file

10. Each task must be written to a file. You may type TASK<cr> to obtain the list of tasks currently defined. Choose file names such as T1.TXT, T2.TXT, etc. since the actual name of the task will be on the disk file. The procedure for each task is as follows (with n=1,2,3,....)

DOC Tn.TXT<cr> ;open a file called Tn.TXT on the current directory

PT task_name<cr> ;write the task called 'task_name'.

Wait for PRINT$ to switch to 0, indicating that the system has finished writing to the disk.

DOC<cr> ;close the Tn.TXT file

11. Produce a list of the current display pages with the PAGE<cr> command and copy it or print it out with the PRTSC<cr> command.

12. For each display page that you have built (and you want to keep):

Display the page.
Type: PD<cr>

This command creates a disk file called name.PIC, where 'name' is the name you have given to the display. (see note on page 9-9 regarding the LIST format).

13. Execute the following RTES commands

DOC S.TXT<cr> ;open a file called S.TXT on the current directory
PS<cr> ;write the scheduled events.

14. Wait for PRINT$ to switch to 0, indicating that the system has finished writing to the disk.

15. Execute the RTES command

DOC<cr> ;close the S.TXT file

16. Exit RTES. (with <Ctrl-Home>)

17. Using the DOS 'delete' or 'erase' command, delete the files OLEX.NAM, OLEX.DAT, OLEX.LNK and OLEX.PAG.

DEL OLEX.NAM<cr>
DEL OLEX.DAT<cr>
DEL OLEX.LNK<cr>
DEL OLEX.PAG<cr>

EDITING THE TEXT FILES

If you have used RTES version C9011xx - or later - to create your .TXT files, you only need to edit S.TXT (skip to 5. below).

1. Using your word processor, read E.TXT as a DOS TEXT FILE, (not a DOCUMENT.)

2. Make the following 'global replace':

[R] changes to RULE<space>
[L] changes to LOOP<space>
[C] changes to SHIFT<space>

[X] changes to EXEC<space>
[E] changes to EXT<space>
[A] changes to ARC<space>

For instance, if the E.TXT file contains the following line

[R]A=B+C

it should now read

RULE A=B+C

3. Save E.TXT on disk as a DOS TEXT FILE (not a DOCUMENT)

4. Using your word processor, read the Tn.TXT as a DOS TEXT FILES, (not DOCUMENTS.), one at the time. Erase "LISTING OF : " from the first line, so that the first line only contains the task name.

For instance :

LISTING OF : AUTOEXEC

should now be

AUTOEXEC

Save each Tn.TXT file after editing, as a DOS TEXT FILE.

5. Using your word processor, read S.TXT as a DOS TEXT FILE, (not a DOCUMENT.)

6. At the front of each point name insert:

SCHED<space>

For instance, if the S.TXT file contains the following line

S1 10:24 OFF

it should now read

SCHED S1 10:24 OFF

7. Save S.TXT on disk as a DOS TEXT FILE (not a DOCUMENT)

LOADING THE KNOWLEDGE BASE IN YOUR NEW RTES.

1. Start up your new - target - RTES.

2. Type the RTES command

FTEXT V.TXT<cr>

This command will load all the variable names and the texts associated with them. While this operation is in progress, the multitasking environment is suspended. The cursor only returns to the beginning of the line, when the operation is completed.

3. Type the RTES command

GET E.TXT<cr>

this command will load all the rules, loop definitions and cascade definitions. While this operation is in progress, the word "READING" is displayed at the top left hand of the screen. Wait for that indicator to disappear before going to the next step.

4. Type the RTES command

GET S.TXT<cr>

this command will load all the schedule definitions. While this operation is in progress, the word "READING" is displayed at the top left hand of the screen. Wait for that indicator to disappear before going to the next step.

5. For every task that you had saved in a Tn.TXT file, type the RTES command

FTASK Tn.TXT<cr>

6. For each screen that you want to import, create a screen with the appropriate name, using the BUILD command.

i.e. BUILD name<cr>

exit the edit mode with the <Home> key, and import the screen using the PL command,

i.e. PL name<cr>

7. You need now to redefine your alert/change-of-state conditions by editing the variable pages as described in the manual.

8. If you are using analog inputs and outputs, you must redefine the analog input TYPEs in the A pages and the ZERO and FULL in the A and Y pages as described in the manual.

9. Your knowledge base conversion is now completed. Save it on disk with the RTES SAVE command or by exiting RTES with <Ctrl-Home>.

SOFTWARE LICENSE AGREEMENT

REAL-TIME EXPERT SYSTEM

RTES-MAX

WinRTES

By using the software purchased under this license, the user becomes a licensee and agrees to the following terms and conditions.

1. The licensee is authorized to make copies of this software for back-up purposes only.
2. The licensee is authorized to make copies of the accompanying manual as required for internal use only.
3. The licensee is authorized to operate the software purchased under this license on a single computer only with not more than one screen and not more than one printer.
4. The licensee will run this software using an operating system obtained under a separate agreement with a supplier other than FAI.
5. FAI guarantees the functionality of this software package to be in accordance with the accompanying manual for a period of 36 month from the date of this agreement. This guarantee is limited to the replacement of the defective original diskette and/or erroneous manual.
6. FAI shall in no case be responsible for any damages resulting directly or indirectly from the use of this software package.

The unauthorized use and/or distribution of this software shall automatically revoke the privileges granted to the licensee by this agreement and shall expose the licensee and unauthorized user(s), jointly and severally, to legal actions by RTS.

Registered serial numbers : L990603 - 616
W2000 - 010

9. Please return registration forms to complete registration process

FAI Designs Inc.
205 Wynford Drive, Suite 1801
Don Mills, Ont.
CANADA
M3C 3P4

(c)1992-2005 FAI Designs Inc.